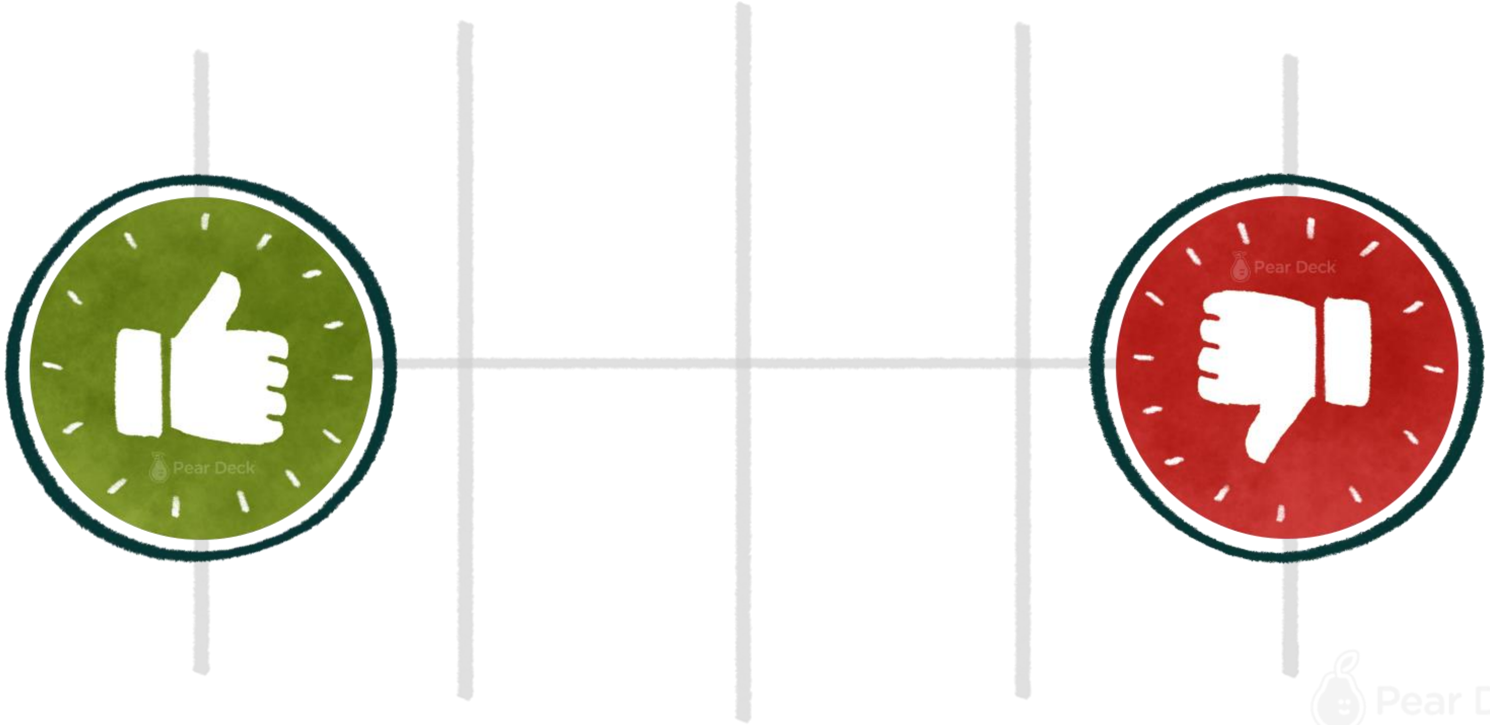
SQL
Session 2

Did you complete the pre-class activity?



Students, drag the icon!

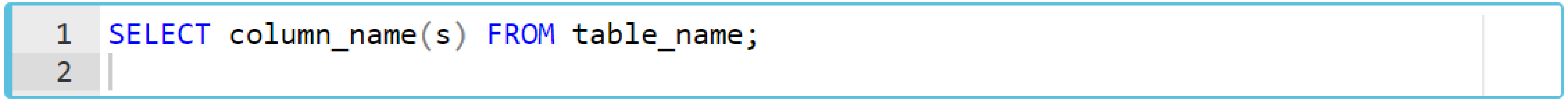
Table of Contents

Structured Query Language (SQL)

SELECT Statement

Introduction

- You can retrieve rows from the columns of the table by using SELECT statement.
- SELECT statement is used with FROM keyword.
- The SELECT statement is used to select data from a database.



```sql
SELECT column_name(s) FROM table_name;

```

SQL

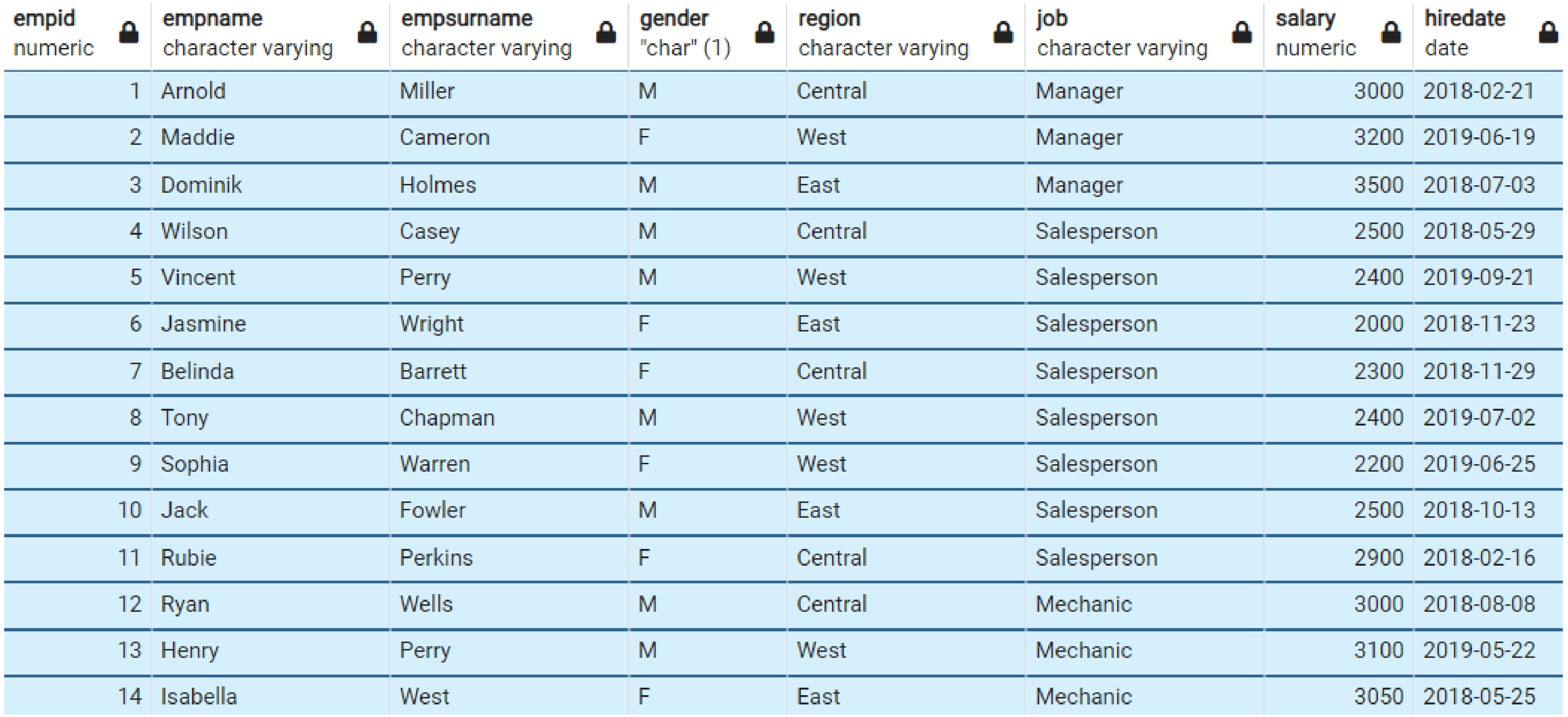
Selecting column /columns/ all columns

SELECT column_name **FROM** table_name**;**

column_name(s)

*

SQL

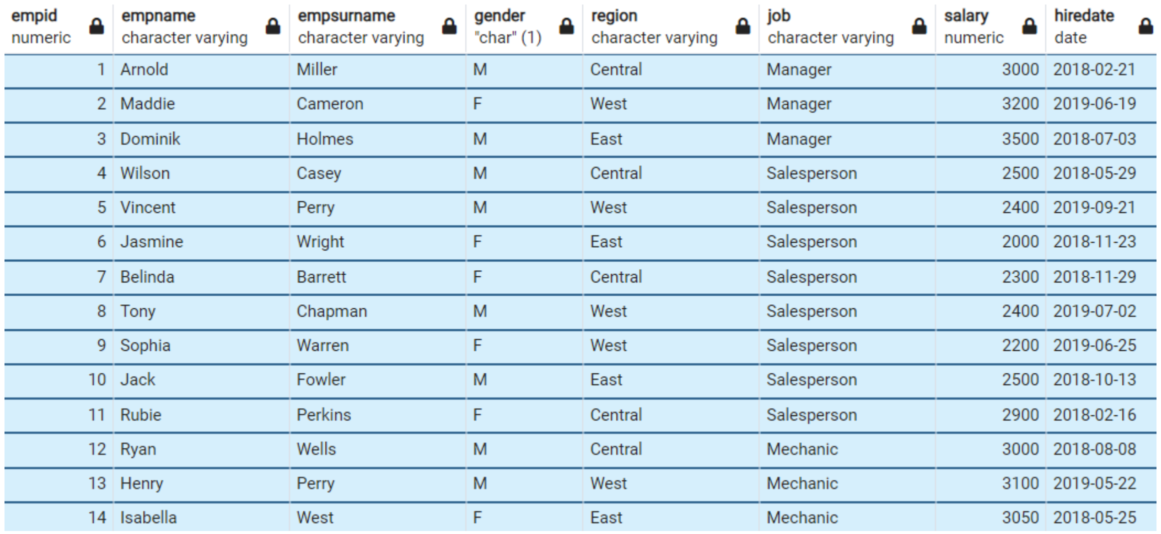


empid	empname	empsurname	gender	region	job	salary	hiredate
numeric	character varying	character varying	"char" (1)	character varying	character varying	numeric	date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

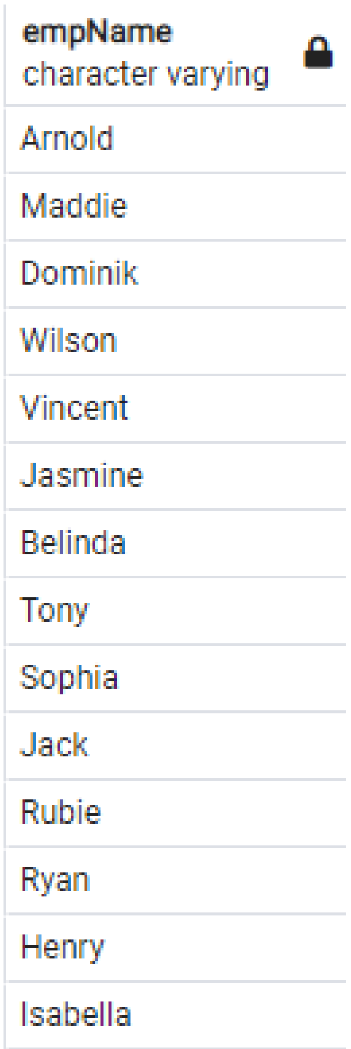
SQL

SELECT empname

FROM employees;



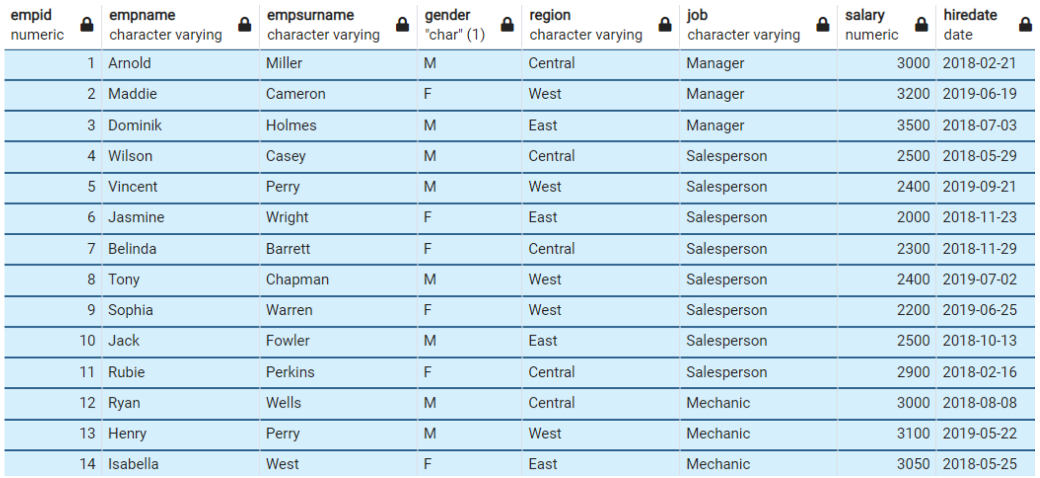
empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25



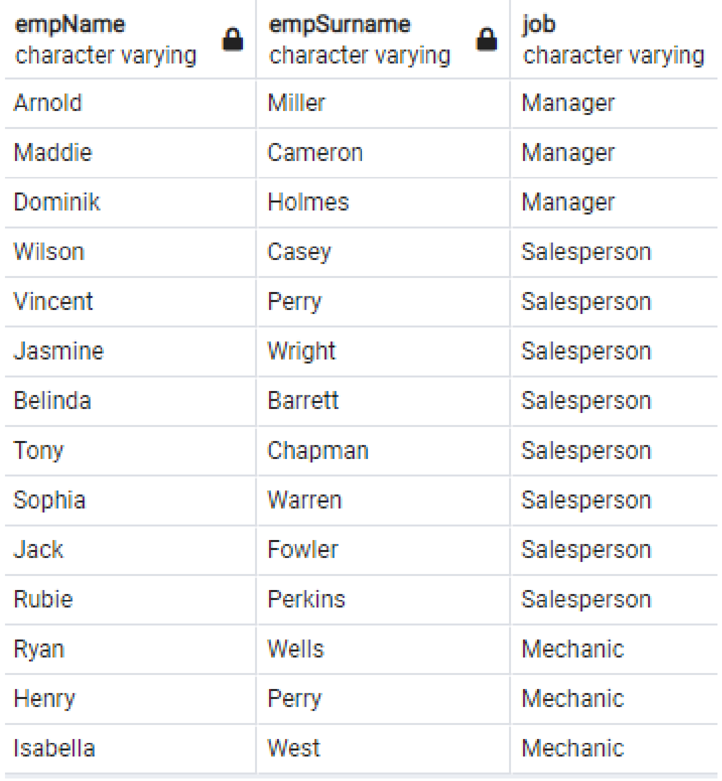
empName character varying
Arnold
Maddie
Dominik
Wilson
Vincent
Jasmine
Belinda
Tony
Sophia
Jack
Rubie
Ryan
Henry
Isabella

SQL

SELECT empname, empsurname, job
FROM employees;



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

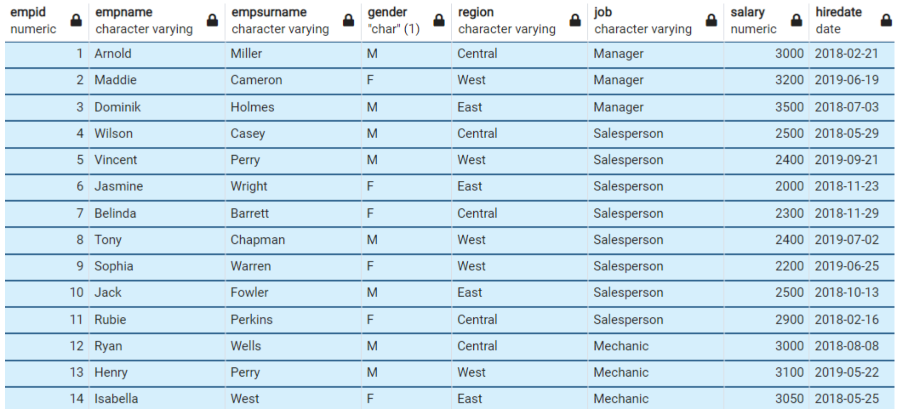


empName character varying	empSurname character varying	job character varying
Arnold	Miller	Manager
Maddie	Cameron	Manager
Dominik	Holmes	Manager
Wilson	Casey	Salesperson
Vincent	Perry	Salesperson
Jasmine	Wright	Salesperson
Belinda	Barrett	Salesperson
Tony	Chapman	Salesperson
Sophia	Warren	Salesperson
Jack	Fowler	Salesperson
Rubie	Perkins	Salesperson
Ryan	Wells	Mechanic
Henry	Perry	Mechanic
Isabella	West	Mechanic

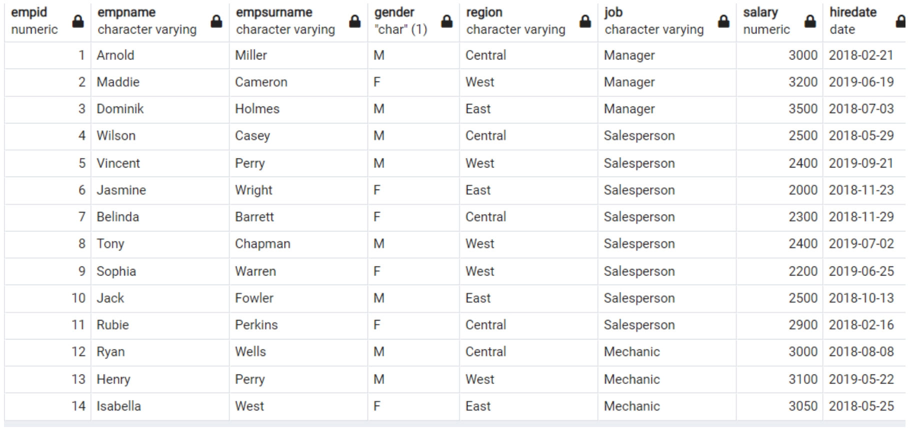
SQL

To retrieve all of the information from your table, an asterisk (*) character can be used after the SELECT

SELECT * **FROM** employees;



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

DISTINCT Clause

SQL

Columns in the tables may often contain some duplicate values, but you may only need the distinct values as a result. In such cases, we use the SELECT statement with the DISTINCT clause.

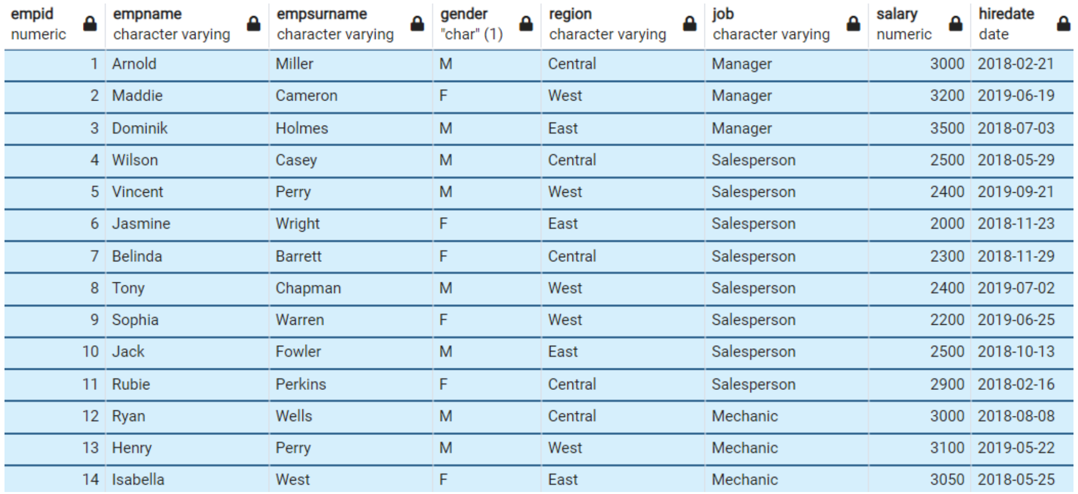
SQL

The SELECT DISTINCT is used to return only distinct (different/unique) values to eliminate duplicate rows in a result set. Here is the syntax of the DISTINCT clause:

```
SELECT DISTINCT column_name(s) FROM table_name;
```

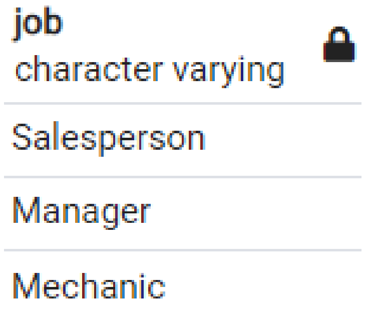
SQL

No Duplicated Rows



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT DISTINCT job
FROM employees;
```

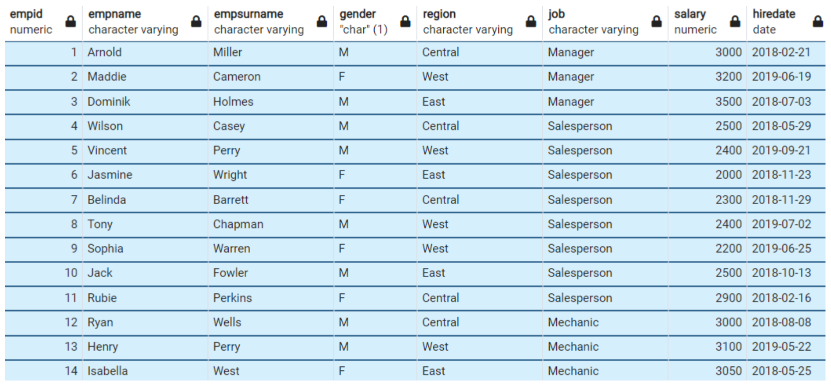


job character varying
Salesperson
Manager
Mechanic

CLARUSWAY©
WAY TO REINVENT YOURSELF

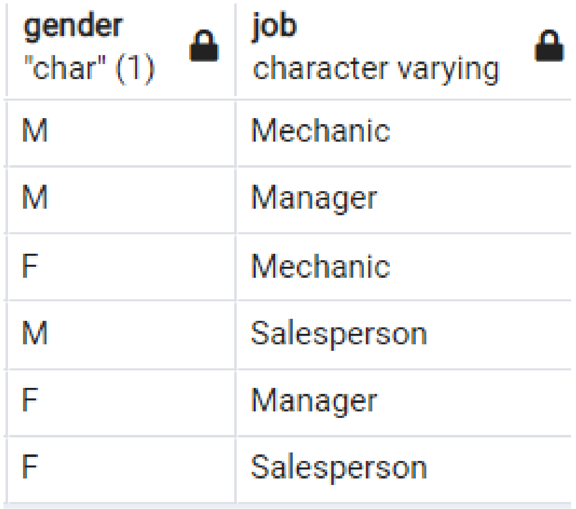
SQL

No Duplicated Rows



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

SELECT DISTINCT gender, job
FROM employees;



gender "char" (1)	job character varying
M	Mechanic
M	Manager
F	Mechanic
M	Salesperson
F	Manager
F	Salesperson

WHERE & LIMIT Clauses

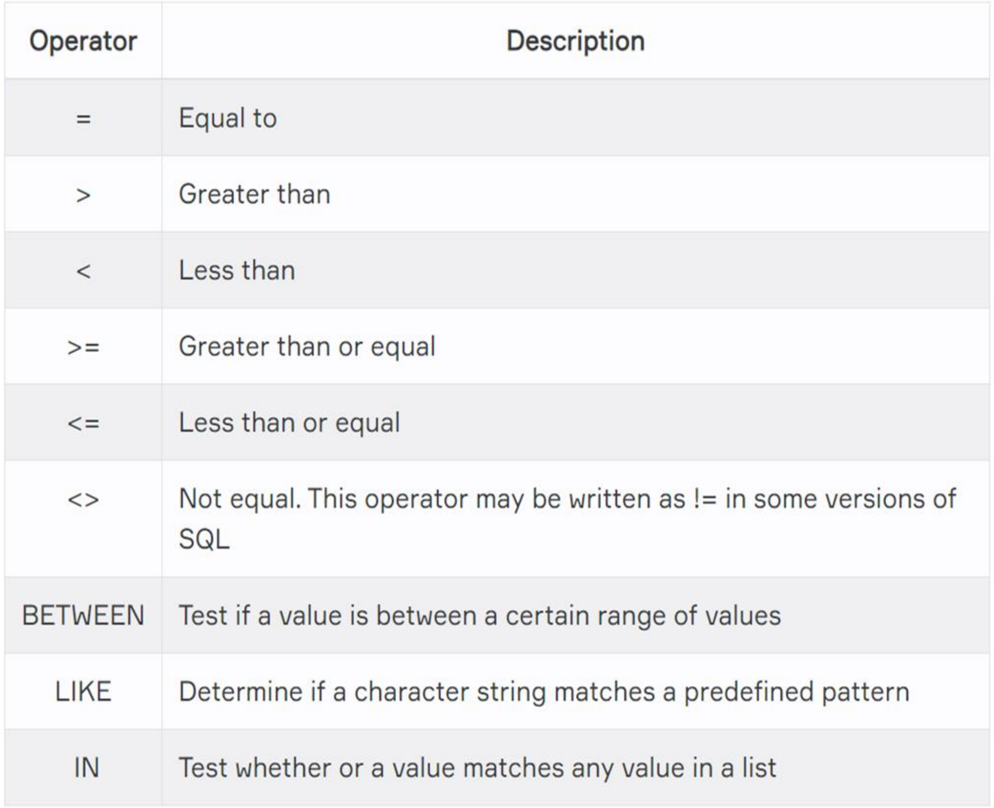
SQL

The WHERE clause is used to filter records.
It allows you to define a specific search condition for the result set returned by a query.

```
SELECT column_name(s)
FROM table_name
WHERE condition(s);
```

SQL

WHERE Clause – Operators

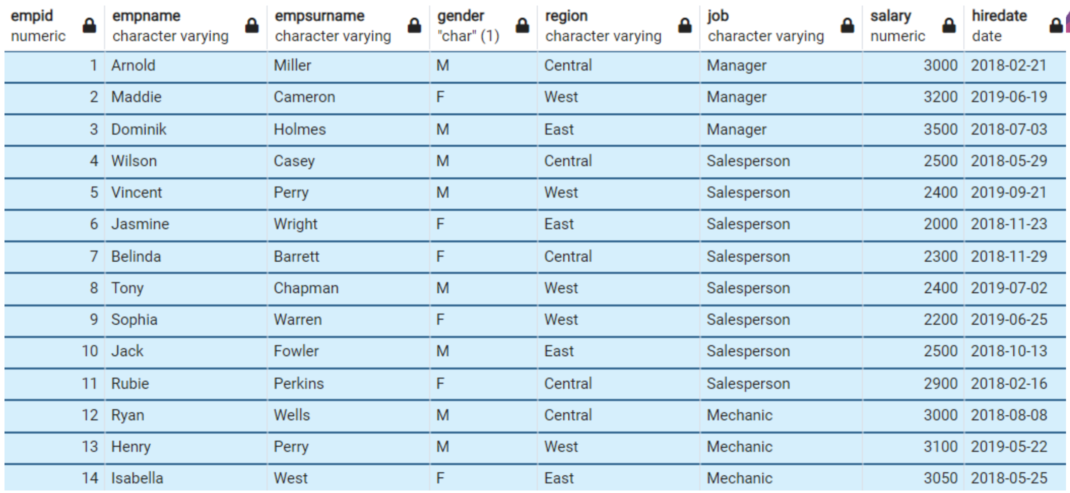


Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. This operator may be written as != in some versions of SQL
BETWEEN	Test if a value is between a certain range of values
LIKE	Determine if a character string matches a predefined pattern
IN	Test whether or a value matches any value in a list

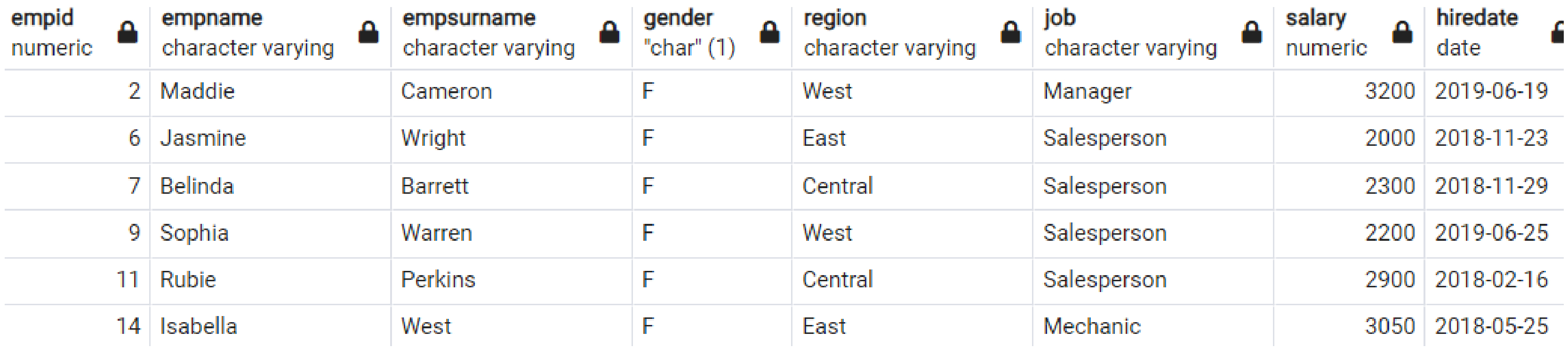
SQL

WHERE Clause

```
SELECT *
FROM employees
WHERE gender='F';
```



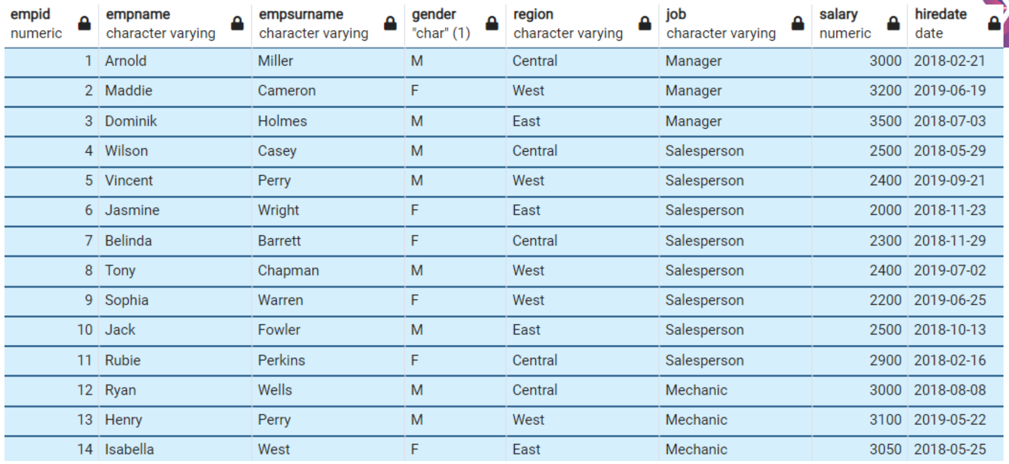
empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

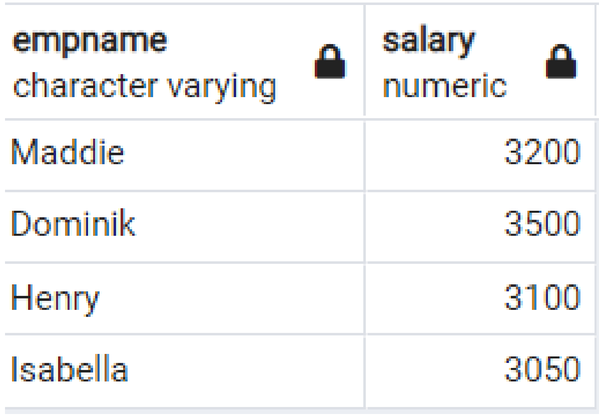
SQL

WHERE Clause



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT empname,salary
FROM employees
WHERE salary>3000;
```



empname character varying	salary numeric
Maddie	3200
Dominik	3500
Henry	3100
Isabella	3050

5 LIMIT Clause

SQL

LIMIT Clause

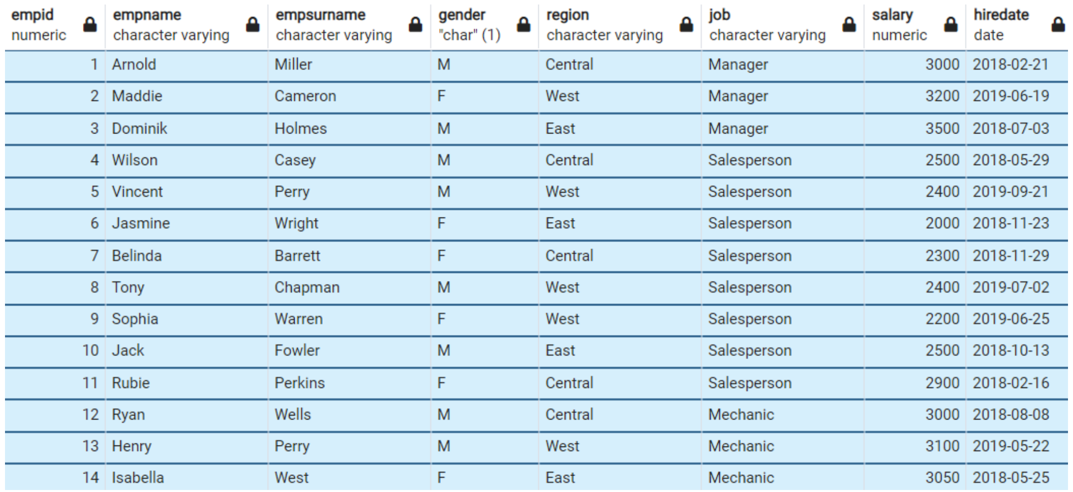
- The LIMIT clause is used to filter records.
- It constrains the number of rows returned by a query.

```
SELECT column_name(s)
FROM table_name
LIMIT number_rows;
```

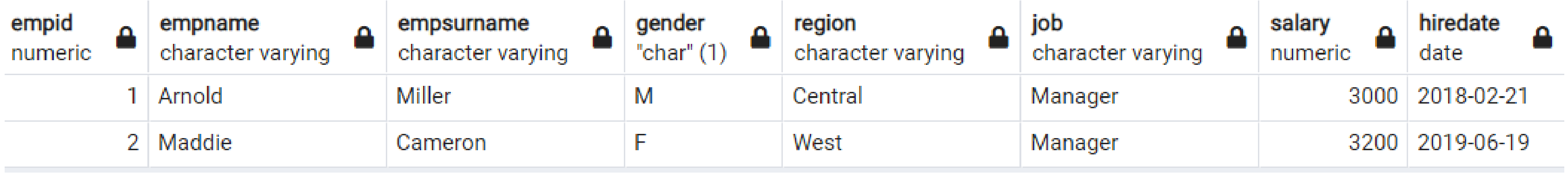
SQL

LIMIT Clause

```
SELECT *
FROM employees
LIMIT 2;
```



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19

SQL

LIMIT Clause

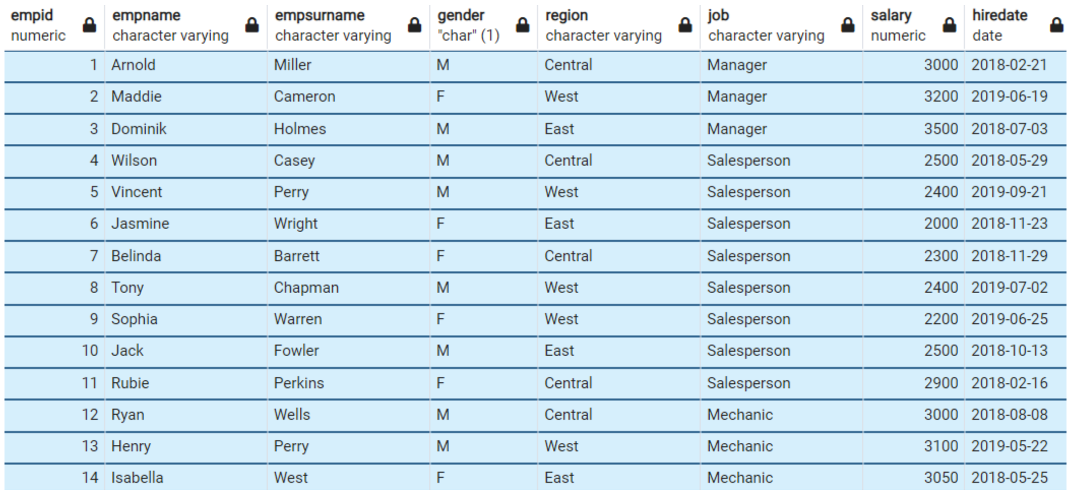
We can also combine LIMIT with WHERE. In that case, LIMIT clause is placed after the WHERE clause.

```
SELECT column_name(s)
FROM table_name
WHERE condition(s);
LIMIT number_rows;
```

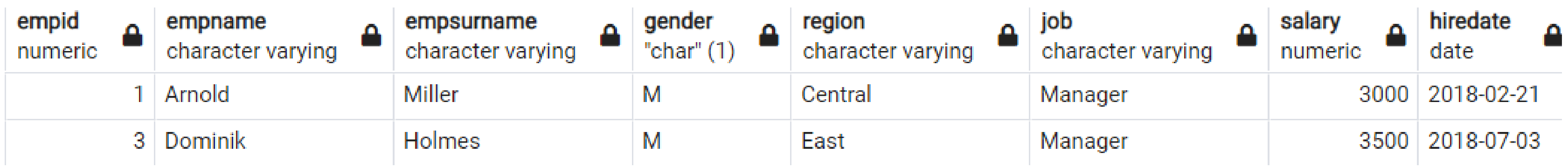
SQL

LIMIT Clause

```
SELECT *
FROM employees
WHERE gender='M'
LIMIT 2;
```



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03

ORDER BY Clause

SQL

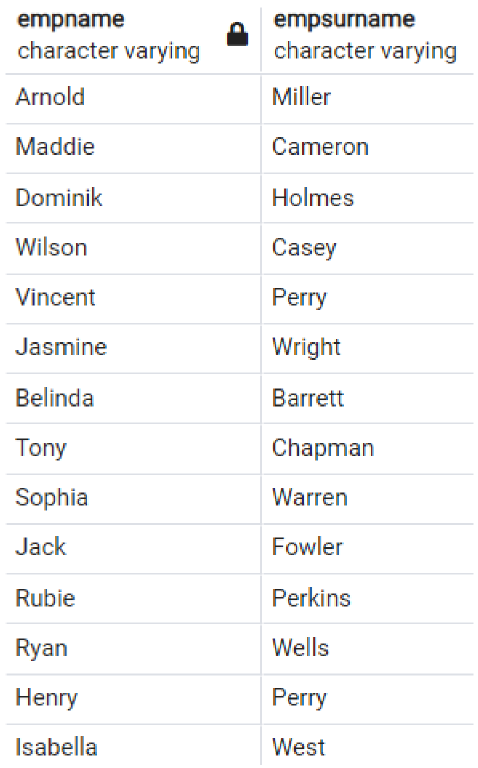
Order By Clause

- In case you want to retrieve data in alphabetical or numeric order, we use ORDER BY keyword.
- By default ORDER BY keyword sorts the records in ascending order.
- Use the keyword DESC to sort the records in descending order. You can also use ASC explicitly to sort the data in ascending order.

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC | DESC
```

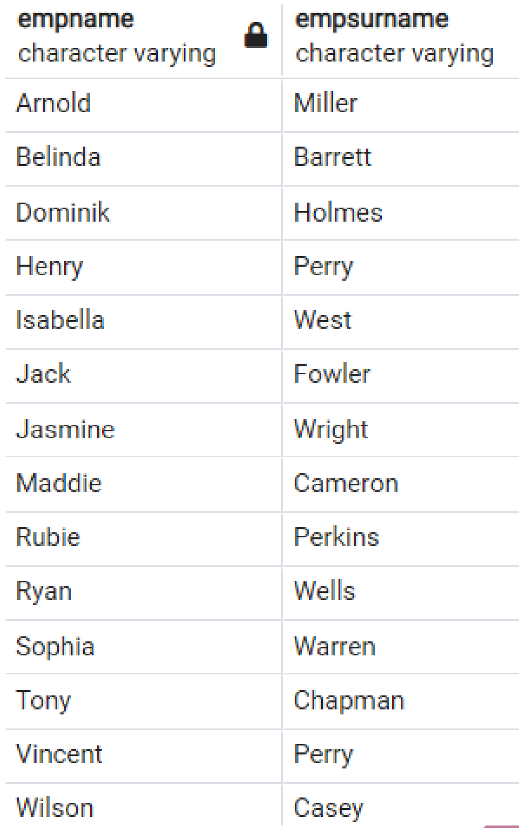
SQL

Order By Clause Ascending Order



empname character varying	empsurname character varying
Arnold	Miller
Maddie	Cameron
Dominik	Holmes
Wilson	Casey
Vincent	Perry
Jasmine	Wright
Belinda	Barrett
Tony	Chapman
Sophia	Warren
Jack	Fowler
Rubie	Perkins
Ryan	Wells
Henry	Perry
Isabella	West

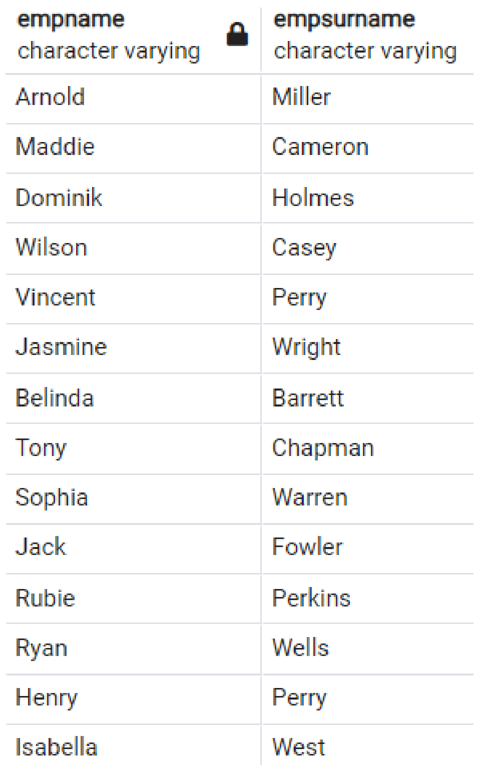
```
SELECT empname,
                    empsurname
FROM employees
ORDER BY empname ASC
```



empname character varying	empsurname character varying
Arnold	Miller
Belinda	Barrett
Dominik	Holmes
Henry	Perry
Isabella	West
Jack	Fowler
Jasmine	Wright
Maddie	Cameron
Rubie	Perkins
Ryan	Wells
Sophia	Warren
Tony	Chapman
Vincent	Perry
Wilson	Casey

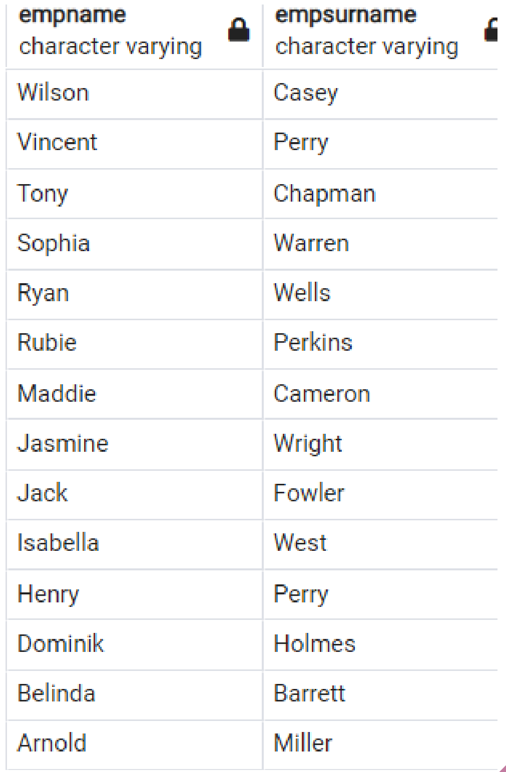
SQL

Order By Clause Descending order



empname character varying	empsurname character varying
Arnold	Miller
Maddie	Cameron
Dominik	Holmes
Wilson	Casey
Vincent	Perry
Jasmine	Wright
Belinda	Barrett
Tony	Chapman
Sophia	Warren
Jack	Fowler
Rubie	Perkins
Ryan	Wells
Henry	Perry
Isabella	West

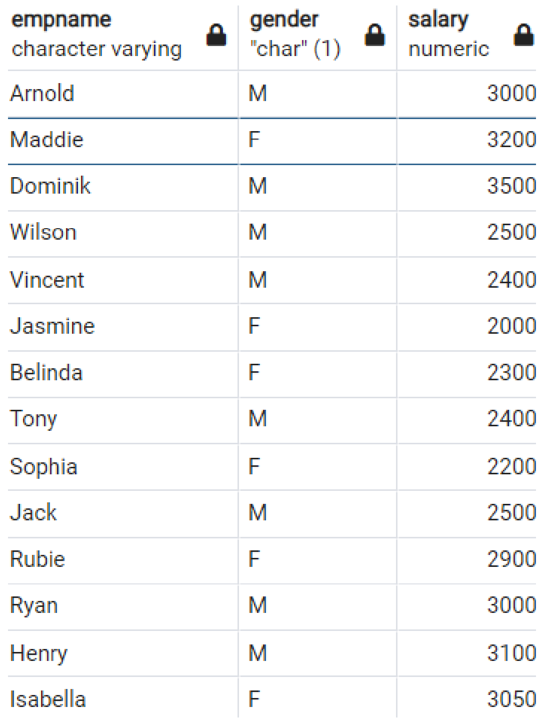
```
SELECT empname,
                    empsurname
FROM employees
ORDER BY empname DESC
```



empname character varying	empsurname character varying
Wilson	Casey
Vincent	Perry
Tony	Chapman
Sophia	Warren
Ryan	Wells
Rubie	Perkins
Maddie	Cameron
Jasmine	Wright
Jack	Fowler
Isabella	West
Henry	Perry
Dominik	Holmes
Belinda	Barrett
Arnold	Miller

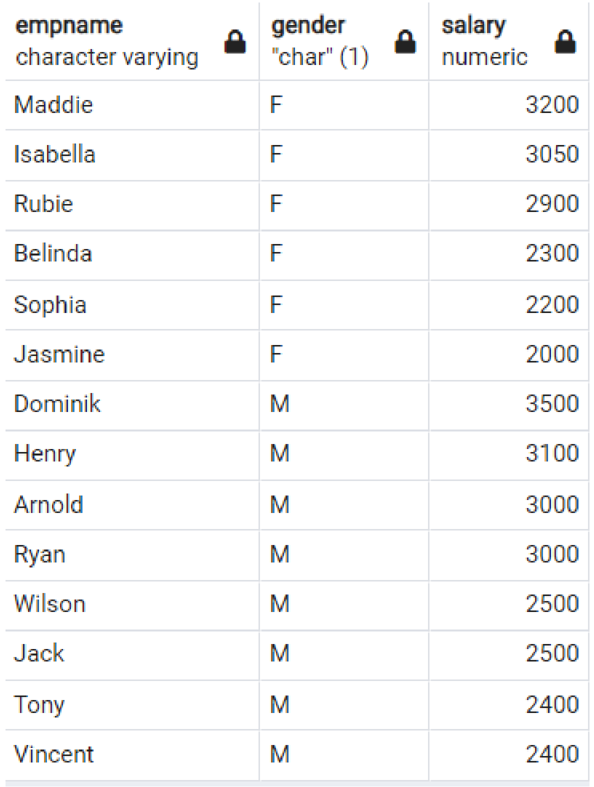
SQL

Order By Clause Multiple Columns



empname character varying	gender "char" (1)	salary numeric
Arnold	M	3000
Maddie	F	3200
Dominik	M	3500
Wilson	M	2500
Vincent	M	2400
Jasmine	F	2000
Belinda	F	2300
Tony	M	2400
Sophia	F	2200
Jack	M	2500
Rubie	F	2900
Ryan	M	3000
Henry	M	3100
Isabella	F	3050

```
SELECT empname,
        gender,
        salary
FROM employees
ORDER BY gender,
          salary DESC;
```

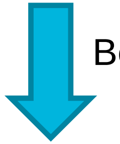


empname character varying	gender "char" (1)	salary numeric
Maddie	F	3200
Isabella	F	3050
Rubie	F	2900
Belinda	F	2300
Sophia	F	2200
Jasmine	F	2000
Dominik	M	3500
Henry	M	3100
Arnold	M	3000
Ryan	M	3000
Wilson	M	2500
Jack	M	2500
Tony	M	2400
Vincent	M	2400

SQL

Order By Clause With WHERE Clause

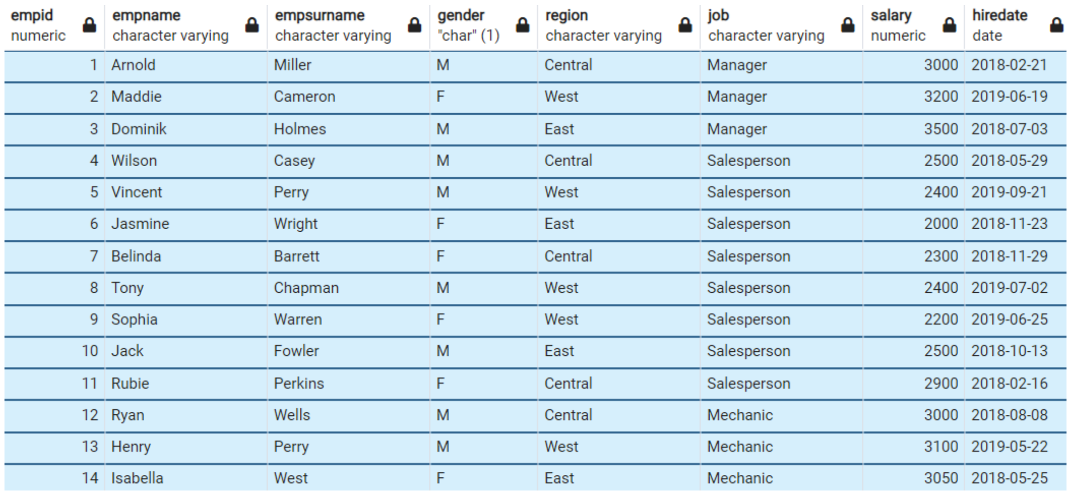
SELECT column_name(s) **FROM** table_name **WHERE** condition(s) **ORDER BY** column_name(s) **ASC | DESC;**



SELECT column_name(s)
FROM table_name
WHERE condition(s)
ORDER BY column_name(s) **ASC | DESC;**

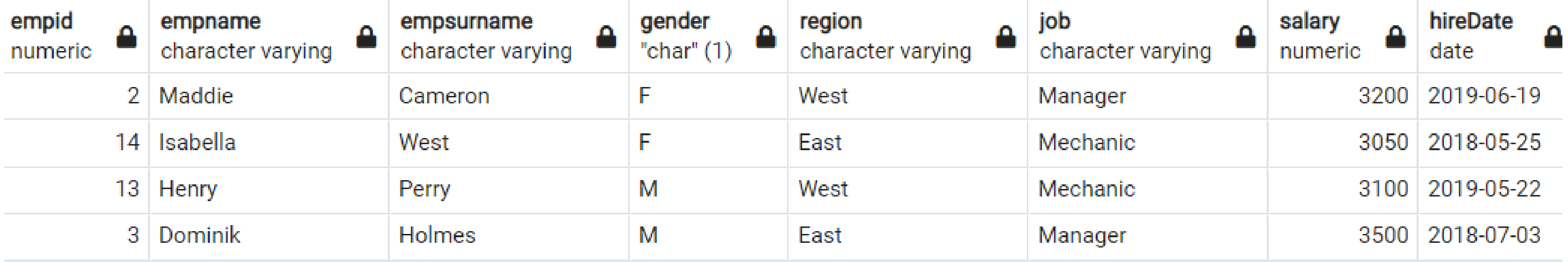
SQL

Order By Clause With WHERE Clause



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

SELECT *
FROM employees
WHERE salary>3000
ORDER BY empname
DESC;



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
14	Isabella	West	F	East	Mechanic	3050	2018-05-25
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03

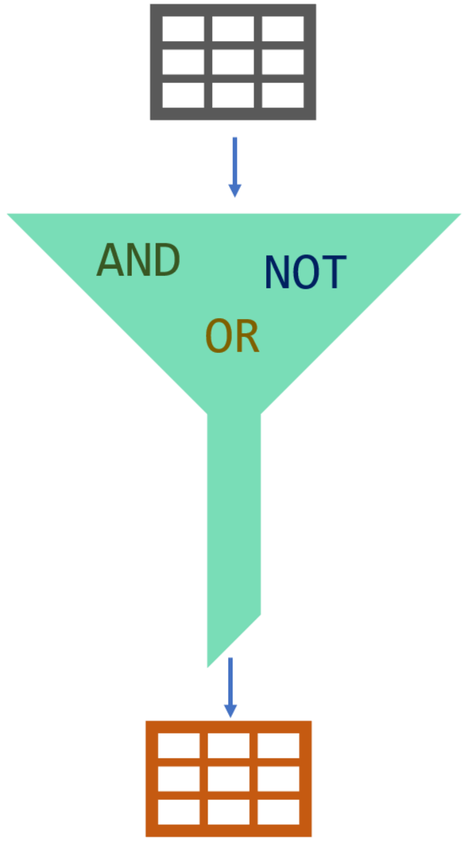
CLARUSWAY©
WAY TO REINVENT YOURSELF

AND, OR & NOT Operators

SQL

AND, OR & NOT Operators

In SQL, **AND, OR & NOT** keywords are called logical operators. Their purposes are filtering the data based on conditions.



SQL

AND Operator

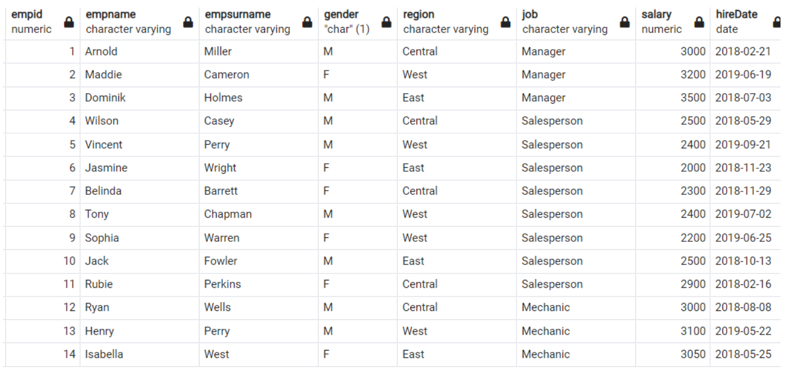
The AND operator is used with the WHERE clause and combines multiple expressions. It returns only those records where both conditions (in WHERE clause) evaluate to True.

Syntax

WHERE left_condition **AND** right_condition

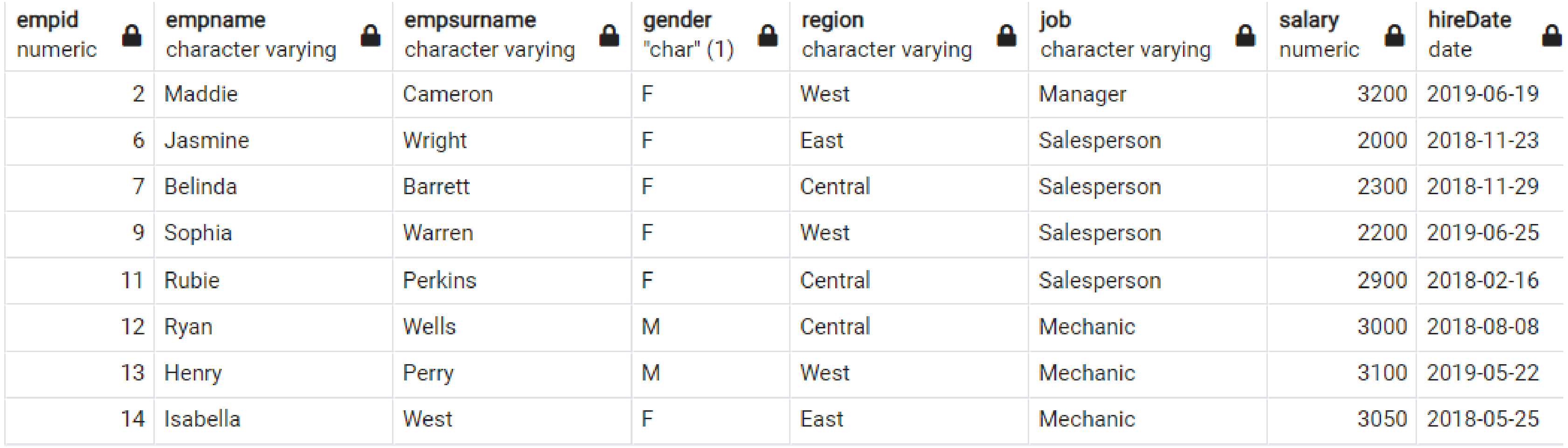
SQL

AND Operator



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT *
FROM employees
WHERE job='Mechanic' OR gender='F';
```



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

SQL

OR Operator

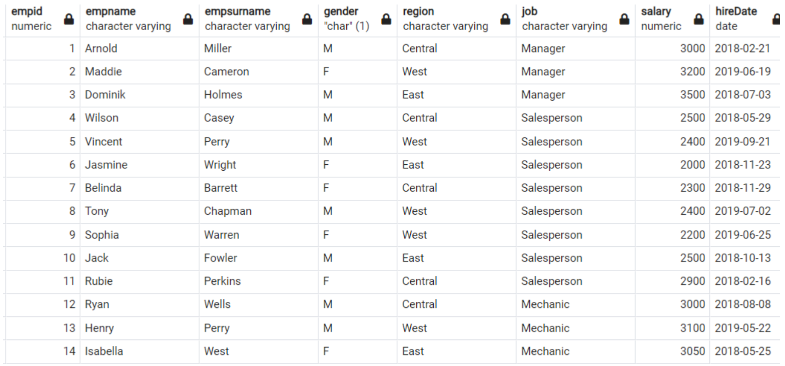
The OR operator is used with the WHERE clause and combines multiple expressions. It displays the record where either one of conditions (in WHERE clause) evaluates to True.

Syntax

WHERE left_condition **OR** right_condition

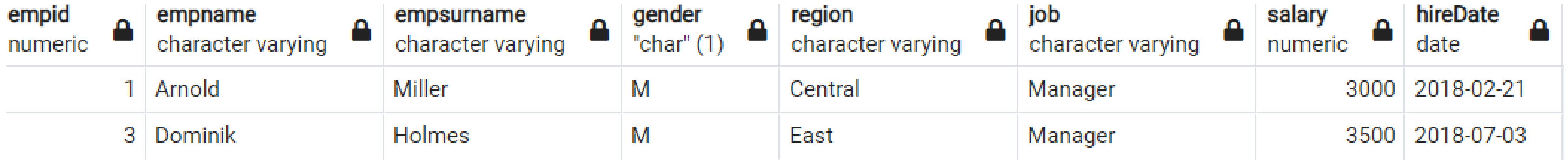
SQL

OR Operator



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT *
FROM employees
WHERE job='Manager' AND gender='M'
```



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03

SQL

NOT Operator

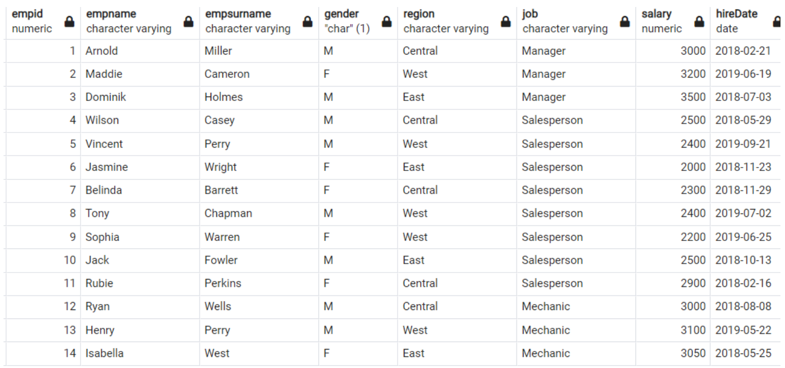
The NOT operator is used to negate a condition in the WHERE clause. NOT is placed right after WHERE keyword. You can use it with AND & OR operators.

Syntax

WHERE NOT first_condition

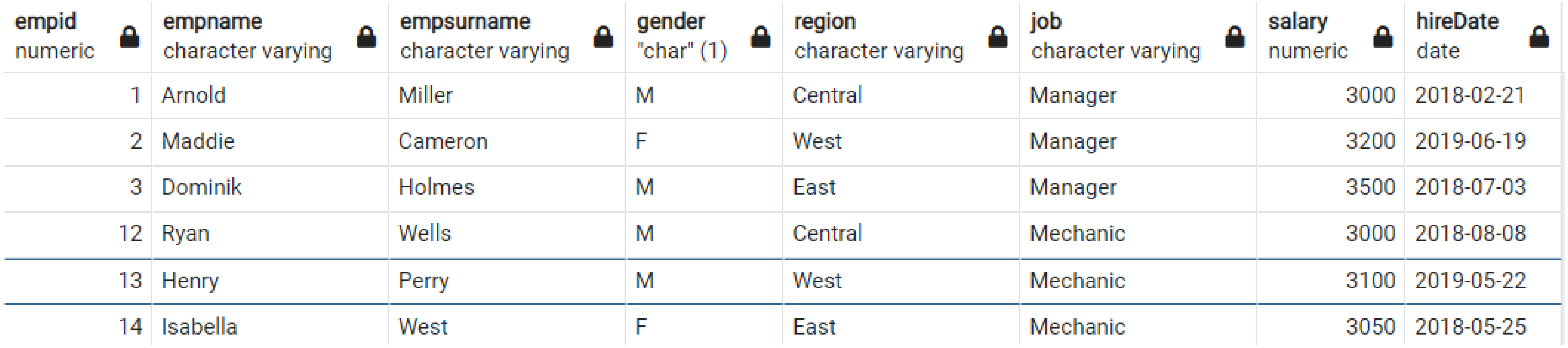
SQL

NOT Operator



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT *
FROM employees
WHERE NOT job='Salesperson'
```



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

BETWEEN OPERATOR

SQL

BETWEEN Operator

The BETWEEN operator is used for comparison in WHERE clauses. It's a comparison operator. You can use it to test if a value is in a range of values. If the value is in the specified range, the query returns all records fallen within that range.

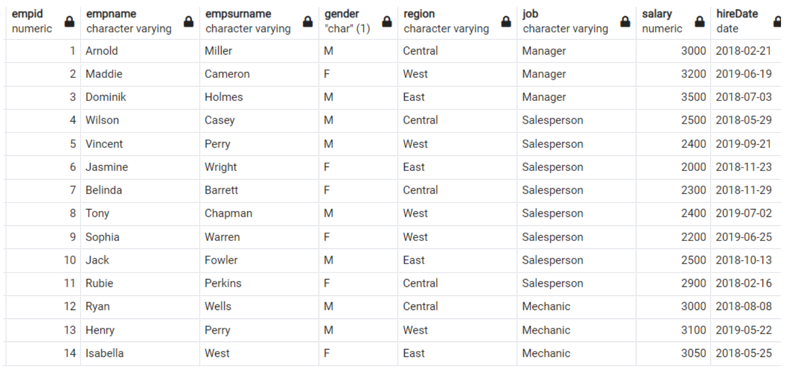
WHERE test_expression **BETWEEN** low_expression **AND** high_expression

=

WHERE test_expression >= low_expression **AND** test_expression <= low_expression

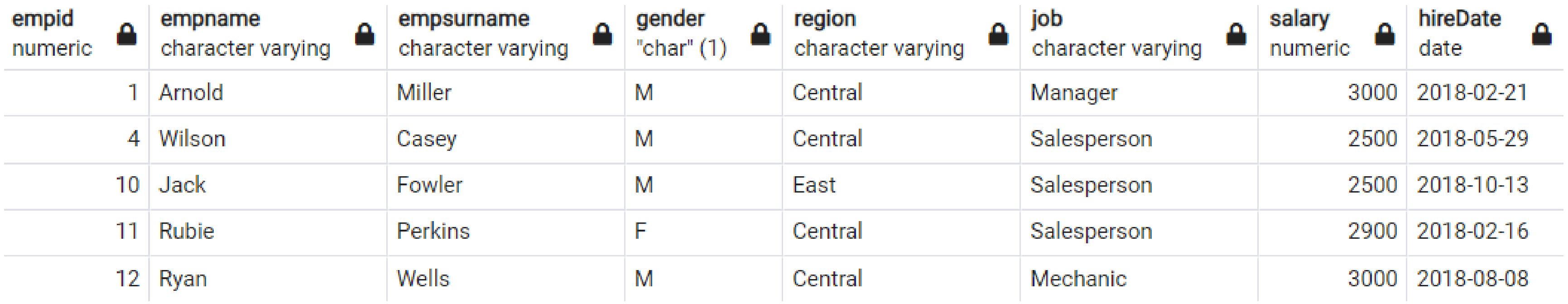
SQL

BETWEEN Operator



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT *
FROM employees
WHERE salary BETWEEN 2500 AND 3000;
```



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hireDate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08

SQL

NOT BETWEEN Operator

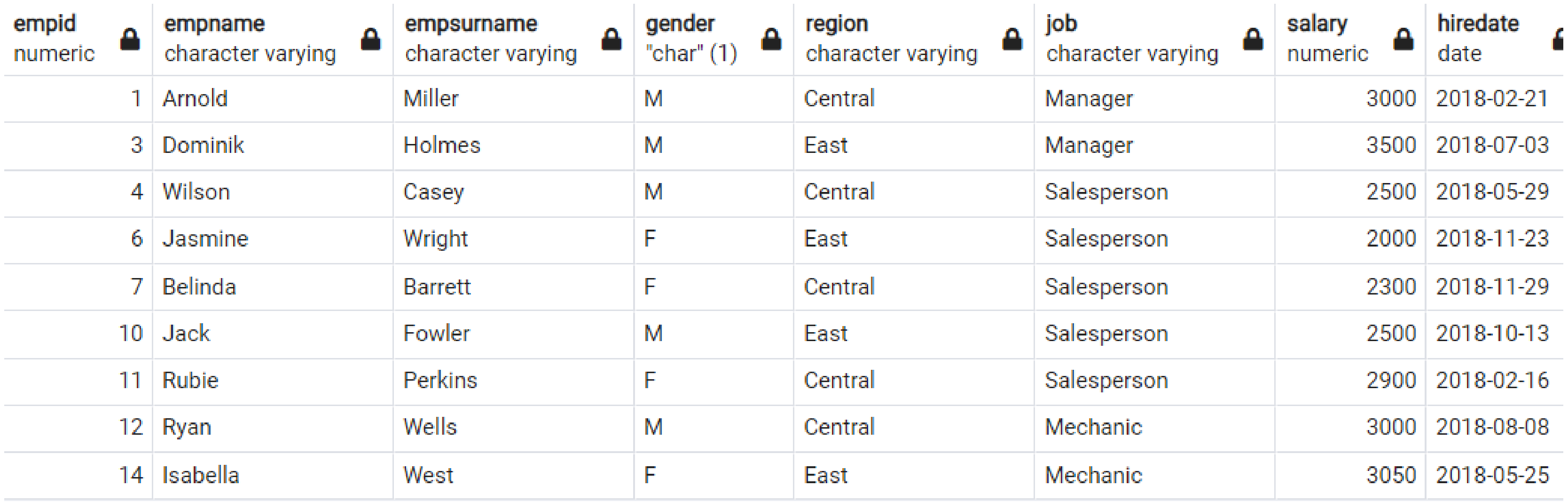
We can use NOT BETWEEN to negate the result of the BETWEEN operator. The following is the syntax:

```
WHERE test_expression NOT BETWEEN low_expression AND high_expression
```

SQL

BETWEEN with Date Example

SELECT *
FROM employees
WHERE hiredate **BETWEEN** '2018-01-01' **AND** '2019-01-01'



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

PRO TIP

Using **BETWEEN** is tricky for datetime! While **BETWEEN** is generally inclusive of endpoints, it assumes the time is at 00:00:00 (i.e. midnight) for **datetime**. So, the end point is exclusive. But, if you have just **date**, then **BETWEEN** behaves as expected.

IN OPERATOR

SQL

IN Operator

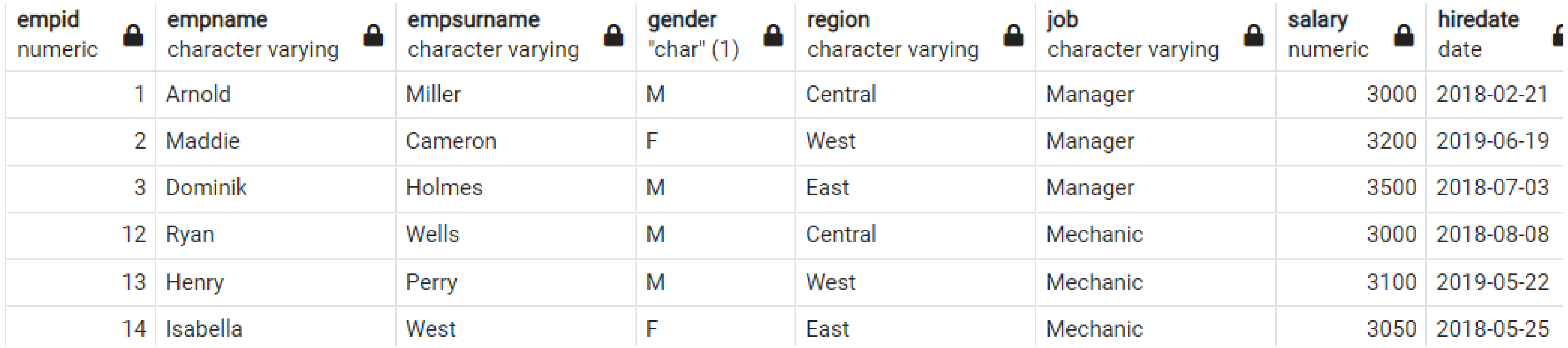
The IN operator is used to determine whether a value matches any value in a list. We use IN operator with WHERE clause.

```
WHERE column_name IN (velue_list)
```

SQL

IN Operator

SELECT *
FROM employees
WHERE job **IN** ('Manager','Mechanic');



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

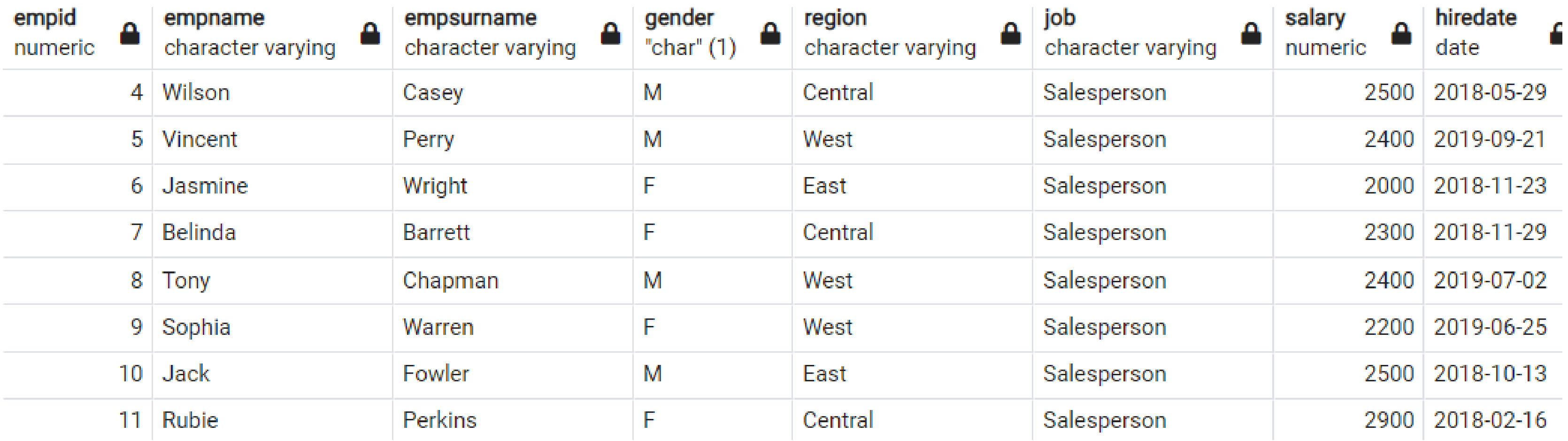
PRO
TIP

If you have a query in which you use many OR operators, consider using the IN operator instead. This will make your query more readable.

SQL

NOT IN Operator

```
SELECT *
FROM employees
WHERE job NOT IN ('Manager','Mechanic');
```



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16

LIKE OPERATOR

SQL

LIKE Operator

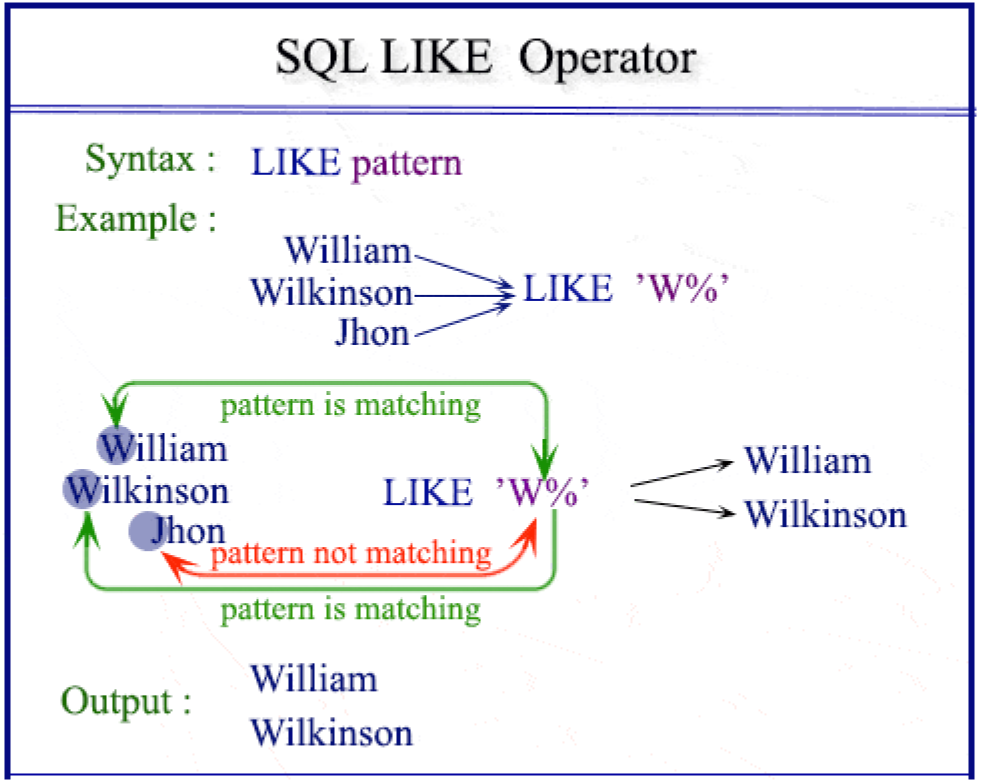
After LIKE keyword, we construct a pattern. SQL provides two special characters for constructing patterns. These are also called wildcards.

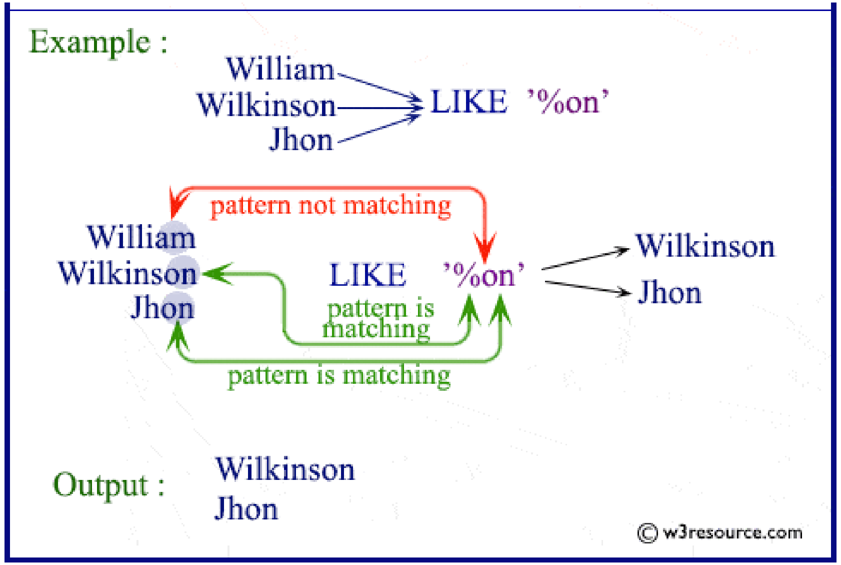
- Percent (%): The % character matches any sequence of zero or more characters.
- Underscore (_): The _ character matches any single character

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE (velue_list)
```

SQL

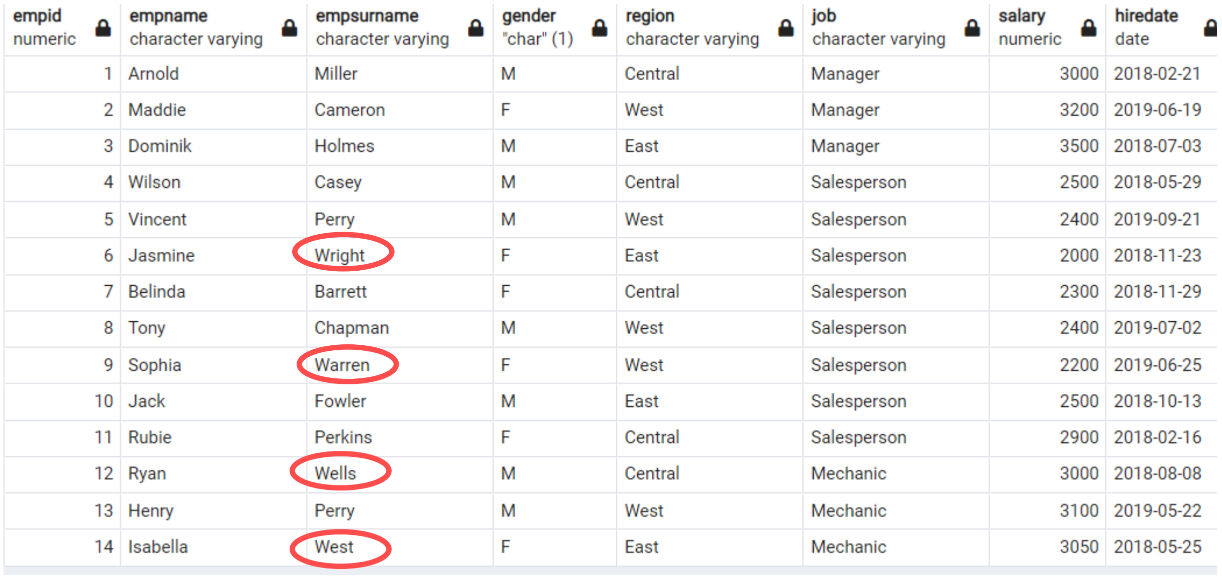
LIKE Operator





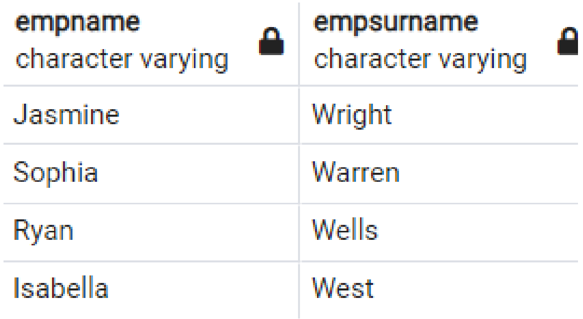
SQL

LIKE Operator



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT empname,empsurname
FROM employees
WHERE empsurname LIKE 'W%';
```

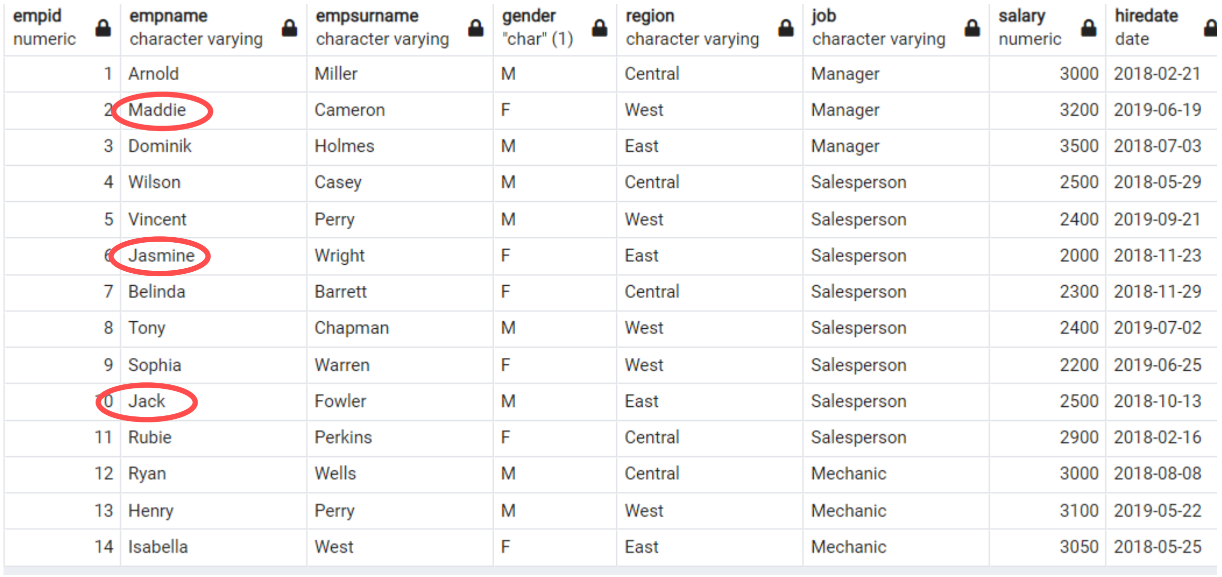


empname character varying	empsurname character varying
Jasmine	Wright
Sophia	Warren
Ryan	Wells
Isabella	West

CLARUSWAY©
WAY TO REINVENT YOURSELF

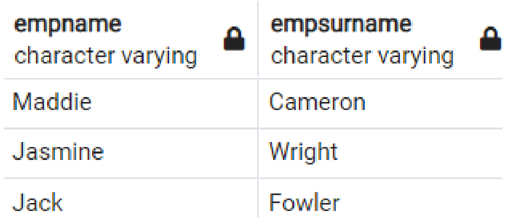
SQL

LIKE Operator



empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	Isabella	West	F	East	Mechanic	3050	2018-05-25

SELECT empname,empsurname
FROM employees
WHERE empname **LIKE** '_a%';



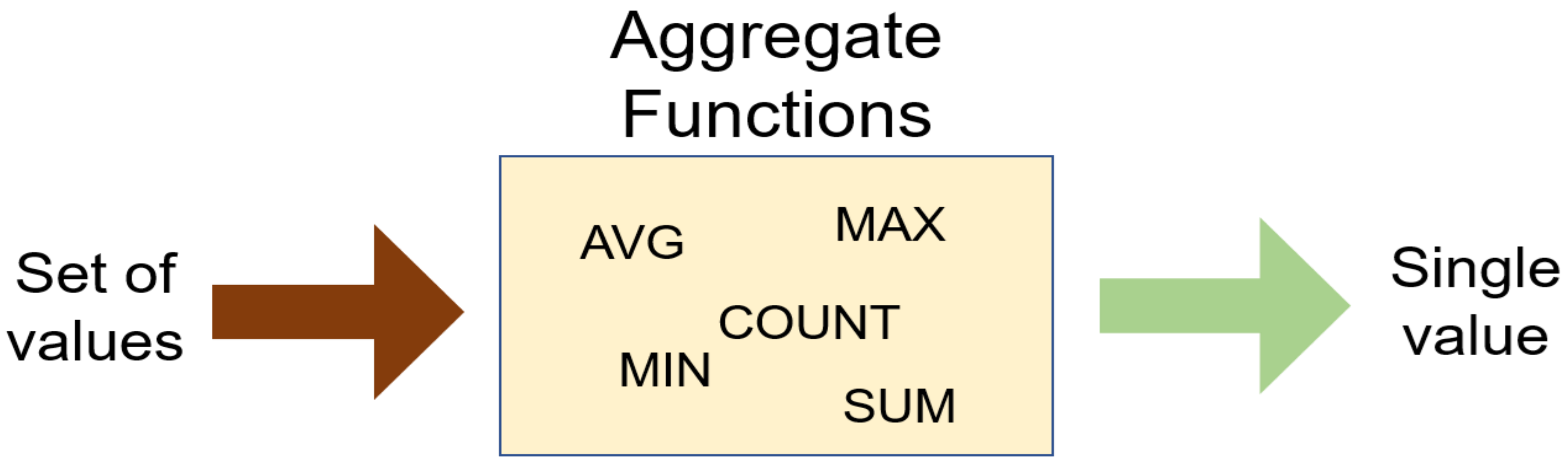
empname character varying	empsurname character varying
Maddie	Cameron
Jasmine	Wright
Jack	Fowler

CLARUSWAY©
WAY TO REINVENT YOURSELF

Aggregate Functions

SQL

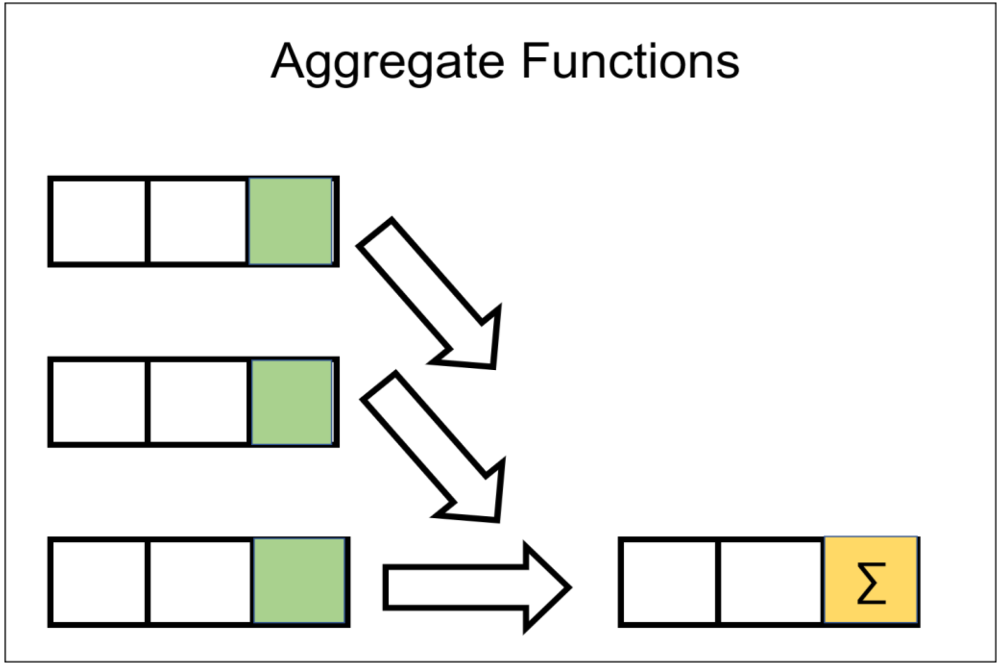
What is an aggregate function?



Aggregate functions are functions that take a collection of values as input and return a single value

SQL

What is an aggregate function?



SUM and AVG → numeric values

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

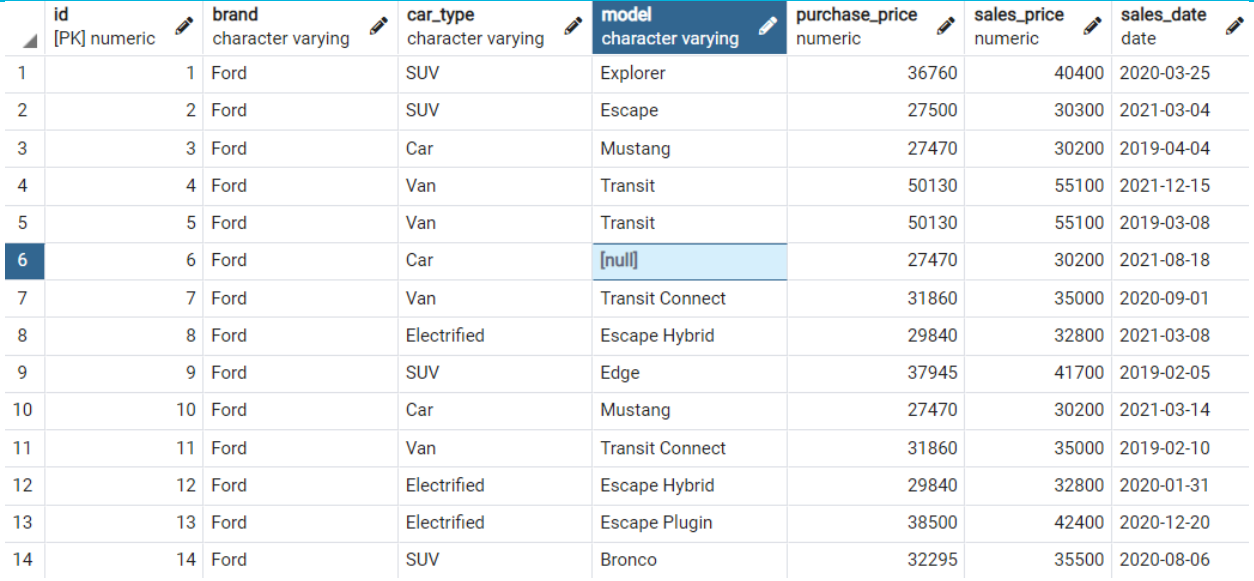
We will learn GROUP BY clause and HAVING clause later.

What is NULL?

SQL

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.



	id [PK] numeric	brand character varying	car_type character varying	model character varying	purchase_price numeric	sales_price numeric	sales_date date
1	1	Ford	SUV	Explorer	36760	40400	2020-03-25
2	2	Ford	SUV	Escape	27500	30300	2021-03-04
3	3	Ford	Car	Mustang	27470	30200	2019-04-04
4	4	Ford	Van	Transit	50130	55100	2021-12-15
5	5	Ford	Van	Transit	50130	55100	2019-03-08
6	6	Ford	Car	[null]	27470	30200	2021-08-18
7	7	Ford	Van	Transit Connect	31860	35000	2020-09-01
8	8	Ford	Electrified	Escape Hybrid	29840	32800	2021-03-08
9	9	Ford	SUV	Edge	37945	41700	2019-02-05
10	10	Ford	Car	Mustang	27470	30200	2021-03-14
11	11	Ford	Van	Transit Connect	31860	35000	2019-02-10
12	12	Ford	Electrified	Escape Hybrid	29840	32800	2020-01-31
13	13	Ford	Electrified	Escape Plugin	38500	42400	2020-12-20
14	14	Ford	SUV	Bronco	32295	35500	2020-08-06

SQL

What is NULL?

- 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

SQL

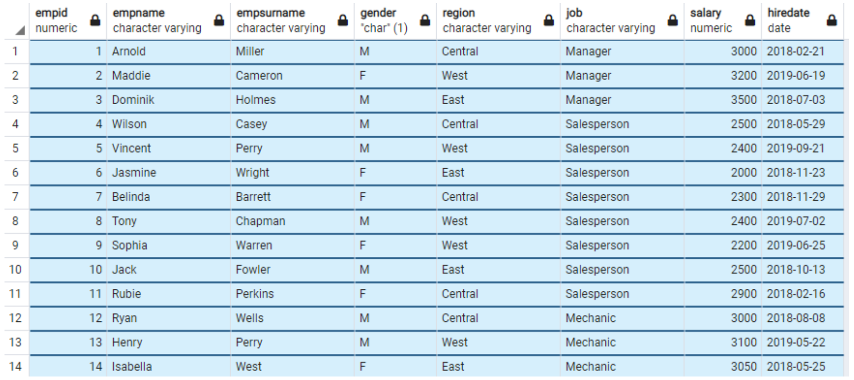
COUNT Function

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

```
SELECT COUNT (column_name)
FROM table_name;
```

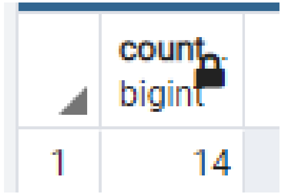
COUNT Function

How many employees does the company have?



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT COUNT (*)
FROM employees;
```



	count bigint
1	14

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.

3

COUNT DISTINCT

COUNT DISTINCT

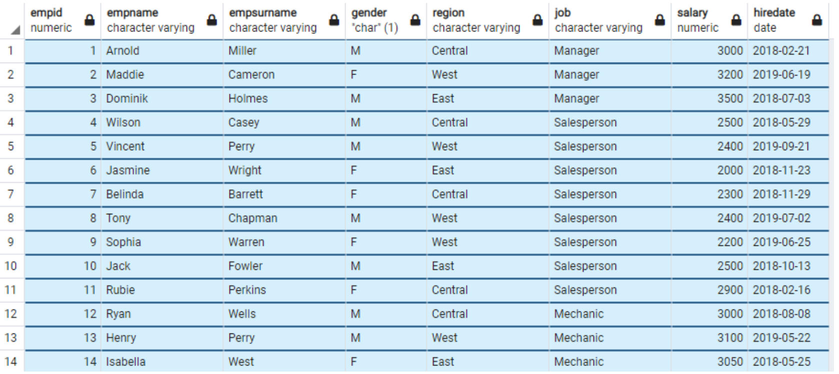
In some cases, we may want unique values. In those cases, we use COUNT DISTINCT function.

Syntax

```
COUNT (DISTINCT column_name)
```

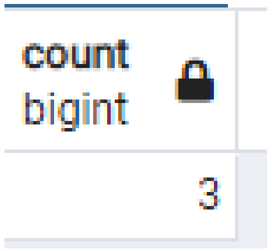
COUNT DISTINCT

How many unique fields are there in the employees table?



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT COUNT (DISTINCT job)
FROM employees;
```



count bigint
3

CLARUSWAY©
WAY TO REINVENT YOURSELF

MIN and MAX

MIN Function

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

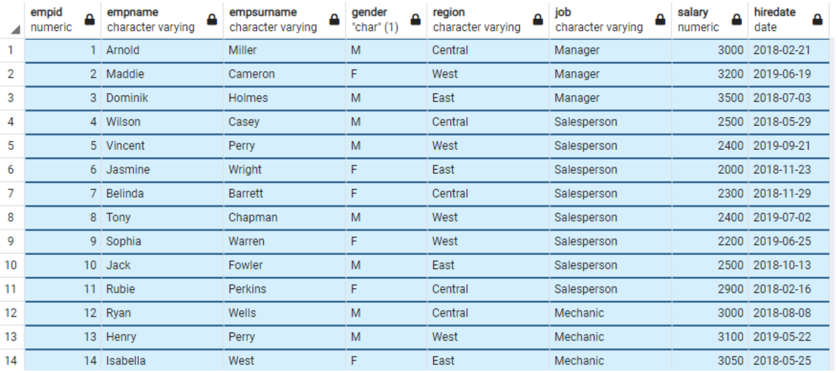
Syntax

```
SELECT MIN (column_name)
FROM table_name;
```

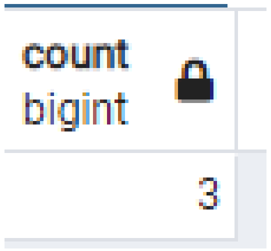
MIN Function

What is the lowest wage in the company?

SELECT MIN (salary)
FROM employees;



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25



count bigint
3

MAX Function

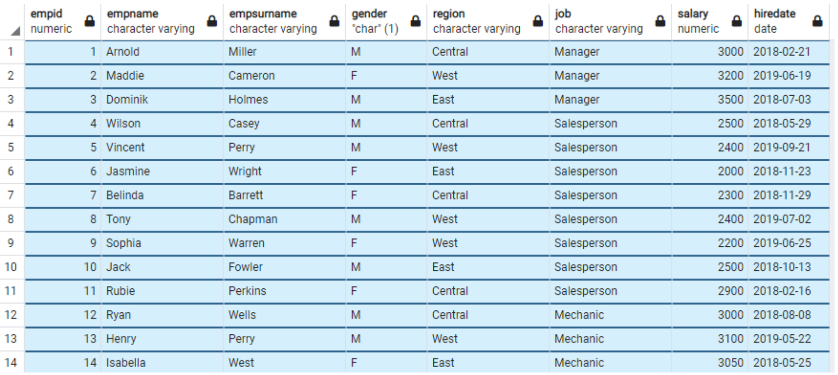
MAX function returns the maximum value in the selected column.

Syntax

```
SELECT MAX (column_name)
FROM table_name;
```

MAX Function

What is the last hired employees's date?



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT MAX (hiredate)
FROM employees;
```

SUM and AVG

SUM Function

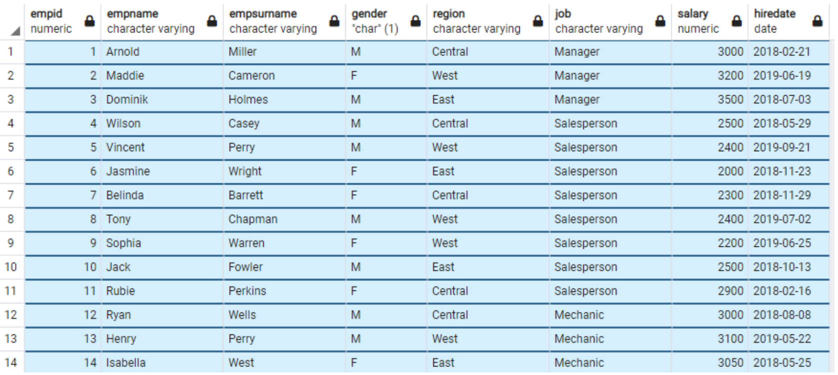
SUM function returns the sum of a numeric column.

Syntax

```
SELECT SUM (column_name)
FROM table_name;
```

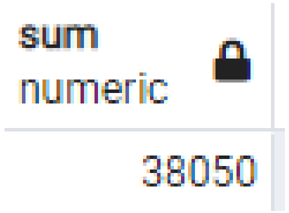
SUM Function

What is total amount salary of the employees?



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT SUM (salary)
FROM employees;
```



sum numeric
38050

CLARUSWAY©
WAY TO REINVENT YOURSELF

2 AVG Function

AVG Function

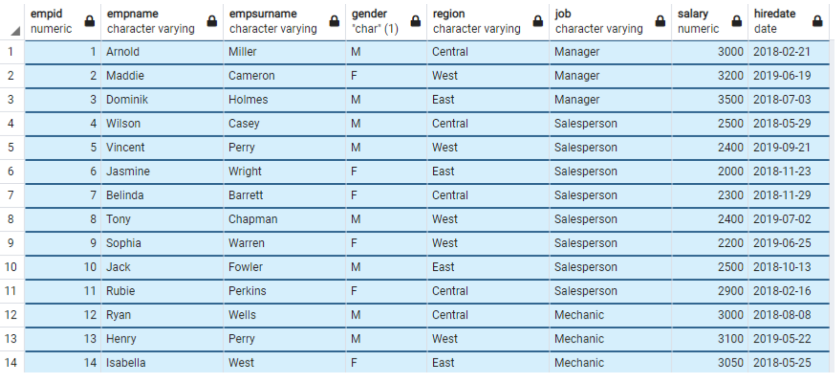
AVG function calculates the average of a numeric column.

Syntax

```
SELECT MAX (column_name)
FROM table_name;
```

AVG Function

What is the average salary of the employees?



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

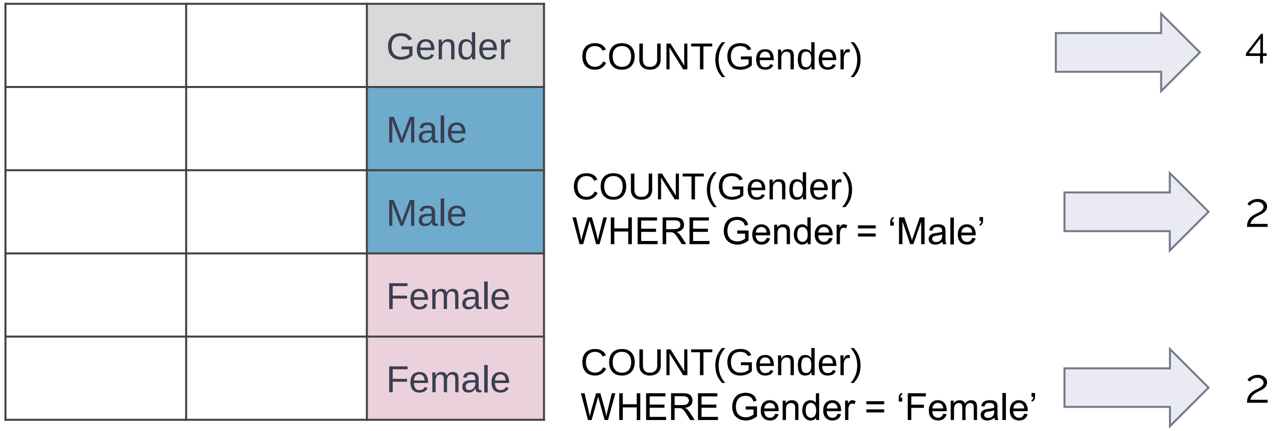
```
SELECT AVG (salary)
FROM employees;
```

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
		Male
		Male
		Female
		Female

COUNT(Gender)	4
COUNT(Gender) WHERE Gender = ‘Male’	2
COUNT(Gender) WHERE Gender = ‘Female’	2

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.

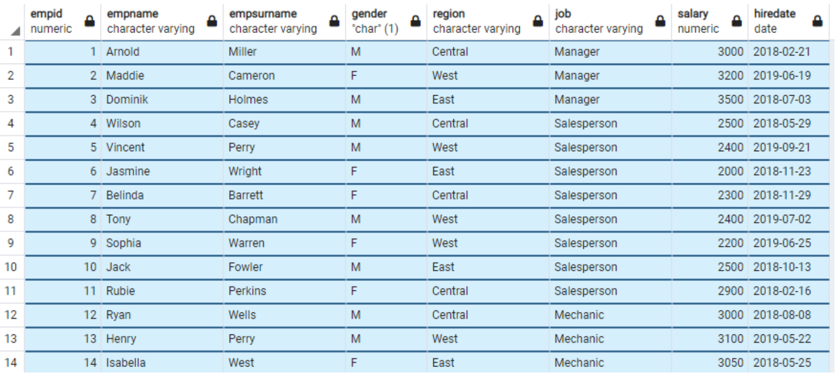
```
SELECT column1,aggregate_function(column2)

FROM tabel_name

GROUP BY column_1;
```

2

GROUP BY with COUNT Function

GROUP BY with COUNT Function

What is the number of employees per gender?

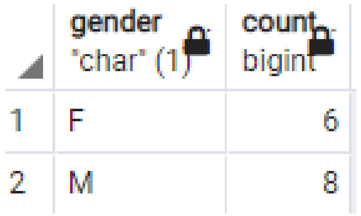


	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

SELECT gender, **COUNT** (gender)

FROM employees

GROUP BY gender;



	gender "char" (1)	count bigint
1	F	6
2	M	8

GROUP BY Clause

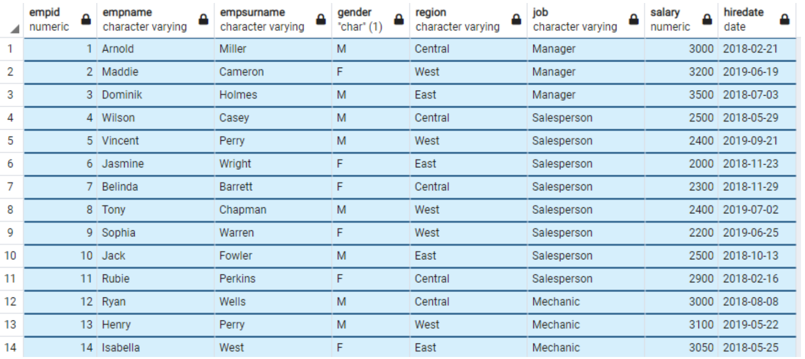
The GROUP BY clause groups results before calling the aggregate function. This allows you to apply aggregate function to groups than the entire query.

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 salesperson broken by gender?



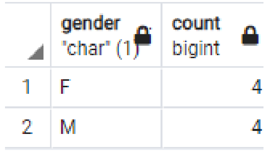
	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

SELECT gender, **COUNT** (job)

FROM employees

WHERE job='Salesperson'

GROUP BY gender;



	gender "char" (1)	count bigint
1	F	4
2	M	4

GROUP BY Clause

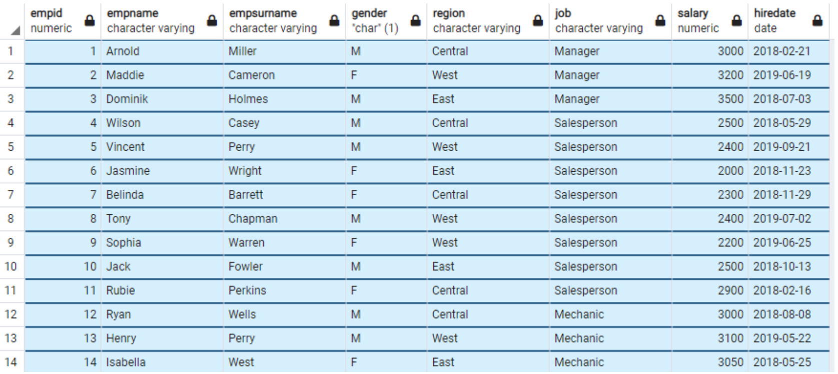
- WHERE clause operates on the data before the aggregation.
- WHERE clause happens before the GROUP BY clause.
- Only the rows that meet the conditions in the WHERE clause are grouped.

3

GROUP BY with MIN&MAX Functions

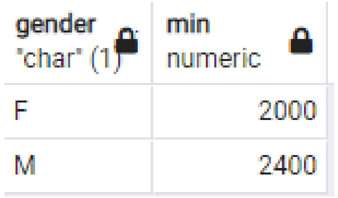
GROUP BY with MIN&MAX Functions

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



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT gender, MIN (salary)
FROM employees
GROUP BY gender;
```

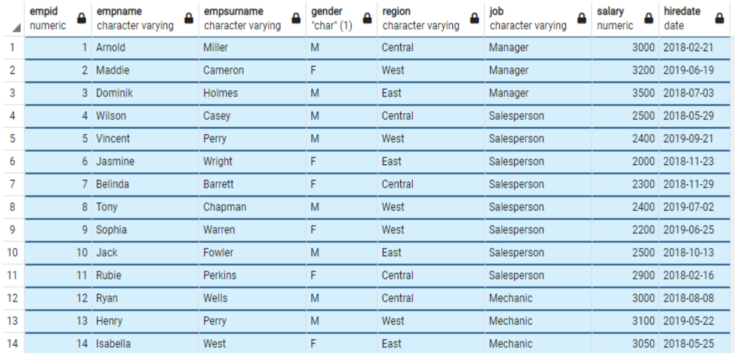


gender "char" (1)	min numeric
F	2000
M	2400

CLARUSWAY©
WAY TO REINVENT YOURSELF

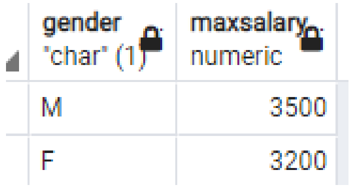
GROUP BY with MIN&MAX Functions

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.



	empid numeric	empname character varying	empsurname character varying	gender "char" (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

```
SELECT gender, MAX (salary) AS maxsalary
FROM employees
GROUP BY gender
ORDER BY maxsalary DESC;
```



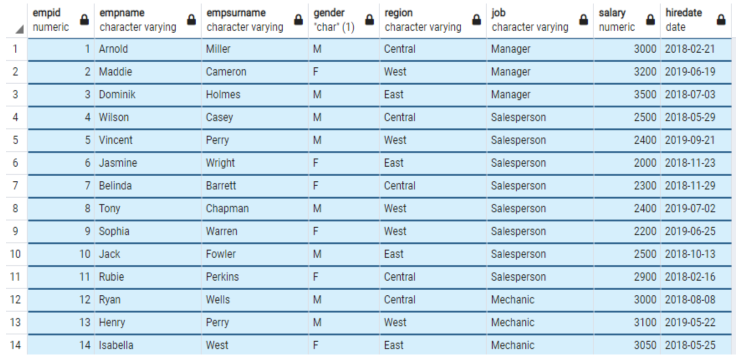
gender "char" (1)	maxsalary numeric
M	3500
F	3200

4

GROUP BY with SUM&AVG Functions

GROUP BY with SUM&AVG Functions

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



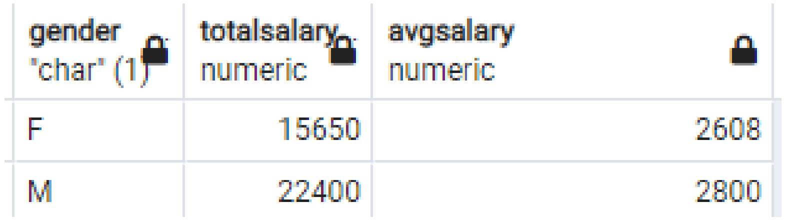
	empid numeric	empname character varying	empsurname character varying	gender 'char' (1)	region character varying	job character varying	salary numeric	hiredate date
1	1	Arnold	Miller	M	Central	Manager	3000	2018-02-21
2	2	Maddie	Cameron	F	West	Manager	3200	2019-06-19
3	3	Dominik	Holmes	M	East	Manager	3500	2018-07-03
4	4	Wilson	Casey	M	Central	Salesperson	2500	2018-05-29
5	5	Vincent	Perry	M	West	Salesperson	2400	2019-09-21
6	6	Jasmine	Wright	F	East	Salesperson	2000	2018-11-23
7	7	Belinda	Barrett	F	Central	Salesperson	2300	2018-11-29
8	8	Tony	Chapman	M	West	Salesperson	2400	2019-07-02
9	9	Sophia	Warren	F	West	Salesperson	2200	2019-06-25
10	10	Jack	Fowler	M	East	Salesperson	2500	2018-10-13
11	11	Rubie	Perkins	F	Central	Salesperson	2900	2018-02-16
12	12	Ryan	Wells	M	Central	Mechanic	3000	2018-08-08
13	13	Henry	Perry	M	West	Mechanic	3100	2019-05-22
14	14	Isabella	West	F	East	Mechanic	3050	2018-05-25

SELECT gender, **SUM** (salary) **AS** totalsalary,

AVG (salary) **AS** avgsalary

FROM employees

GROUP BY gender;

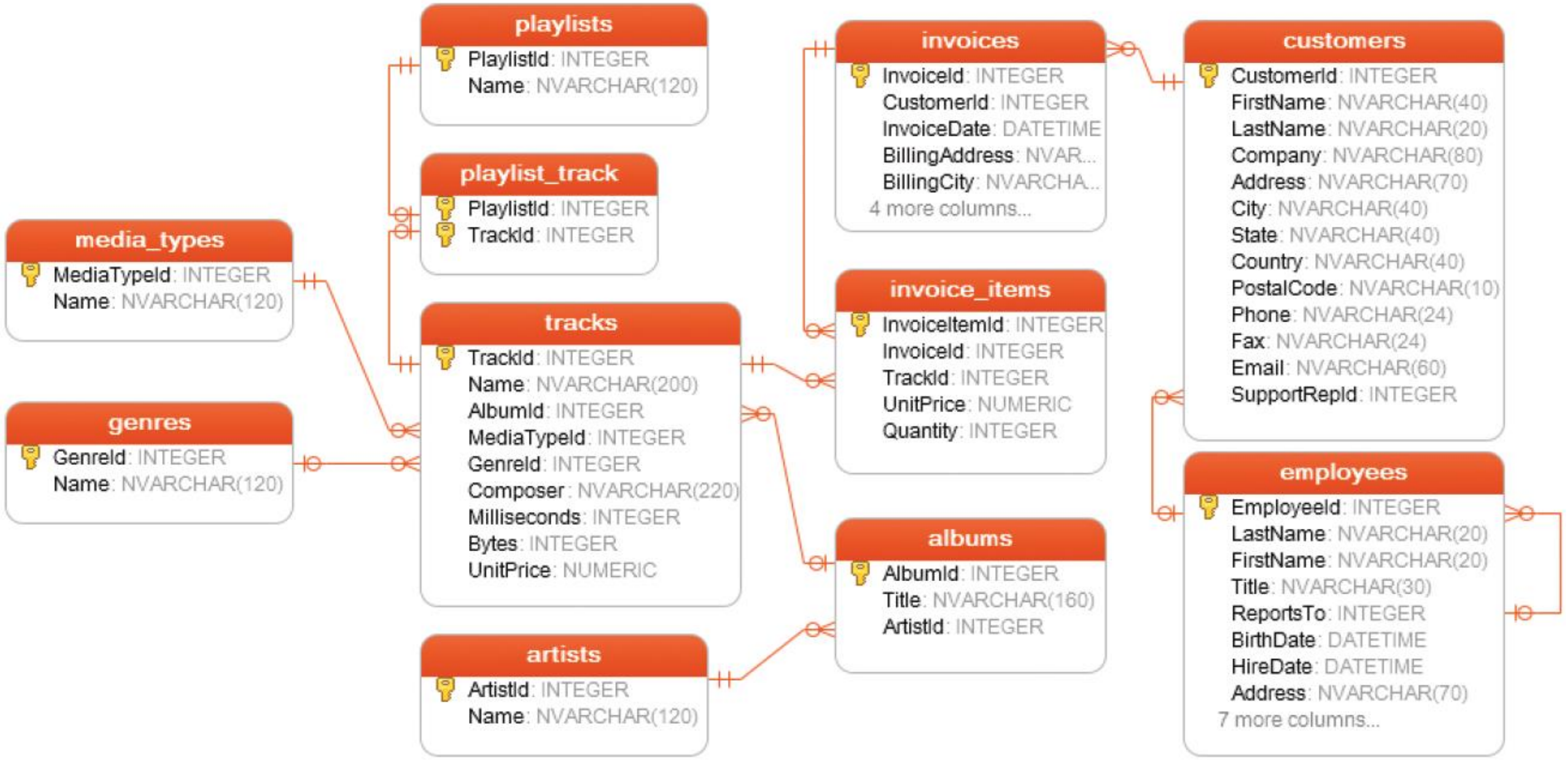


gender 'char' (1)	totalsalary numeric	avgsalary numeric
F	15650	2608
M	22400	2800

CLARUSWAY©
WAY TO REINVENT YOURSELF

Query Time

Entity Relationship Diagram



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...

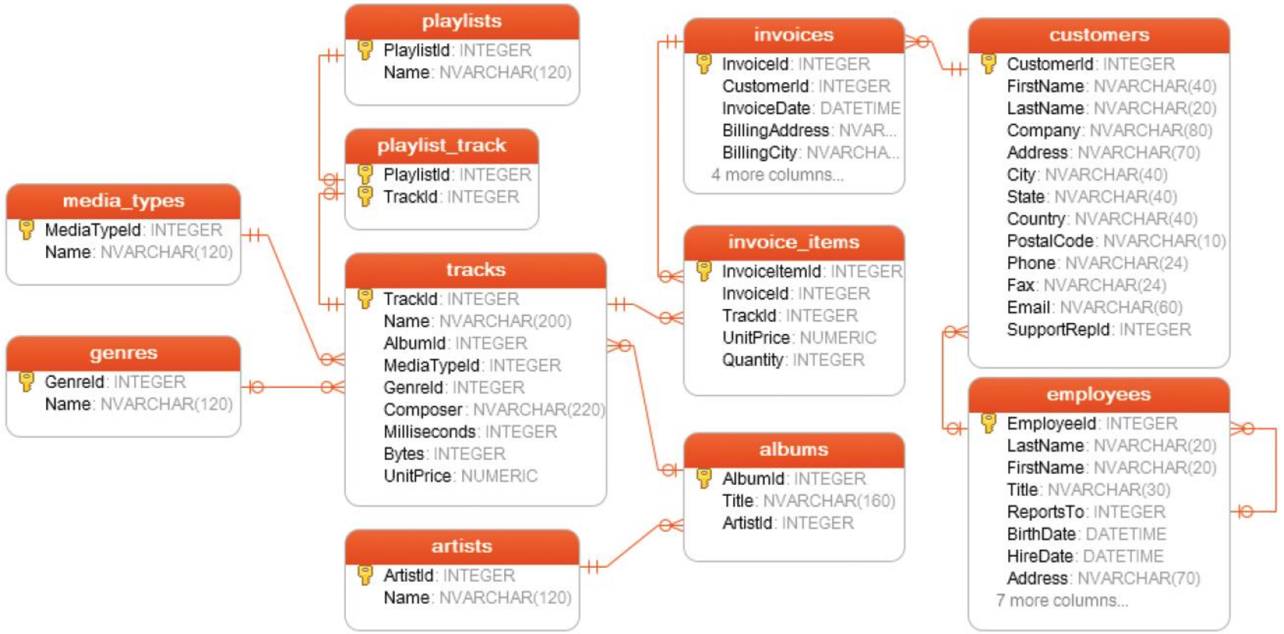
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

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

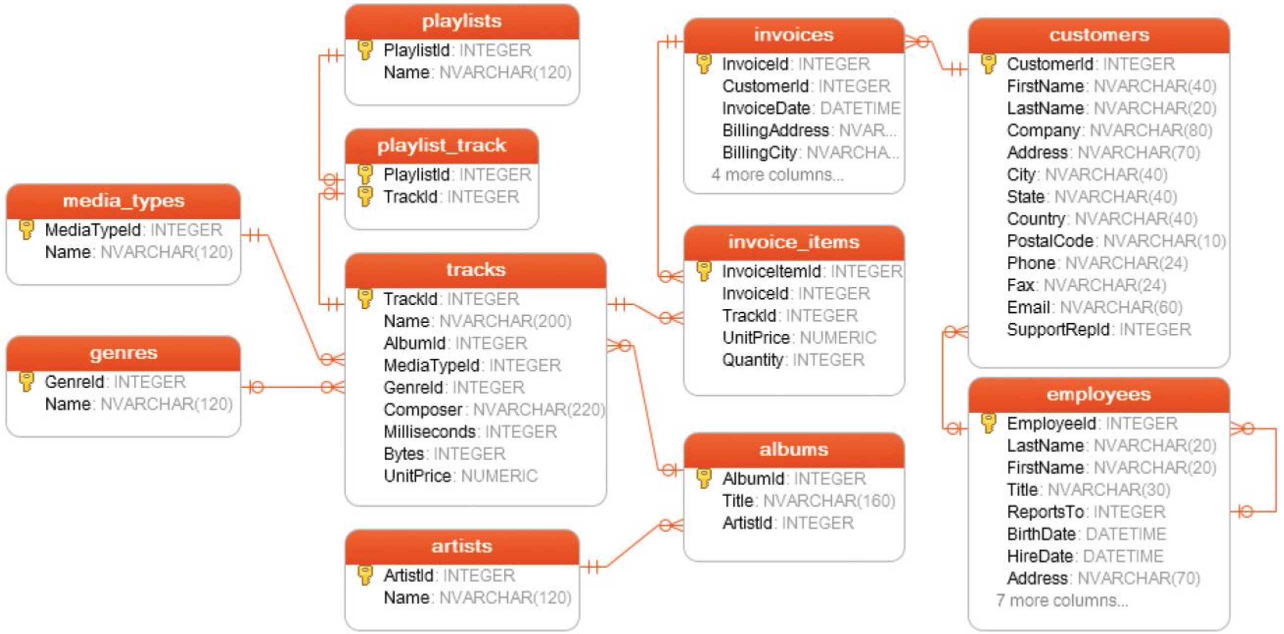
employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

Write a query that returns the track name using tracks table.

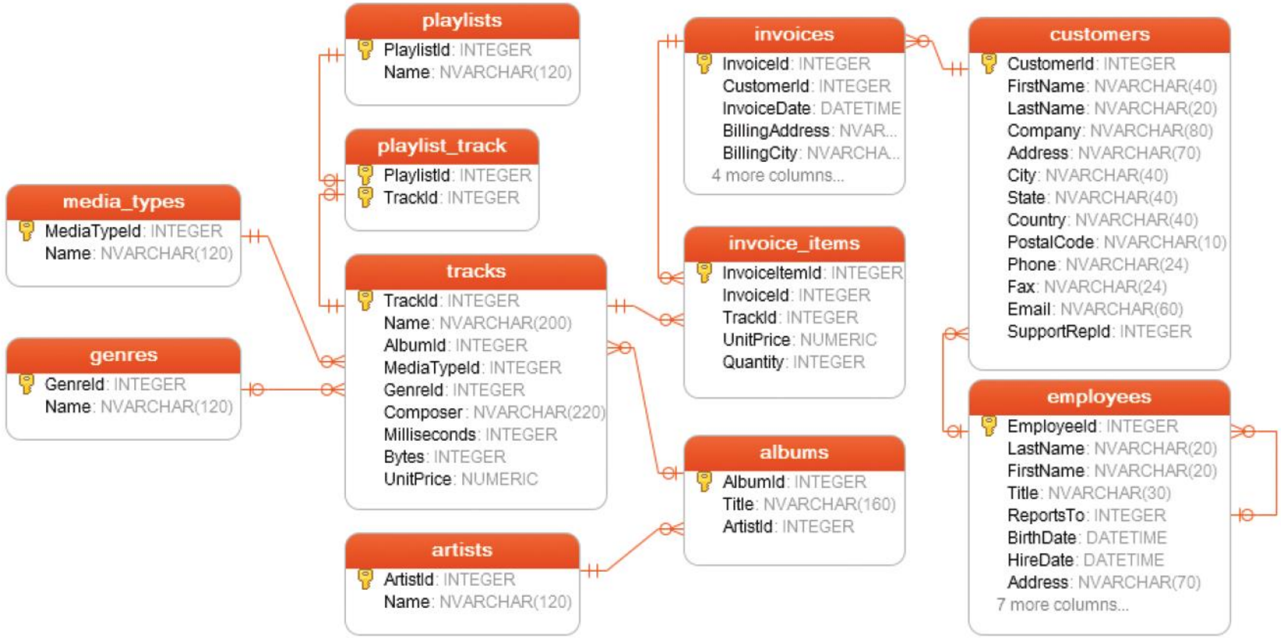


CLARUSWAY©
WAY TO REINVENT YOURSELF

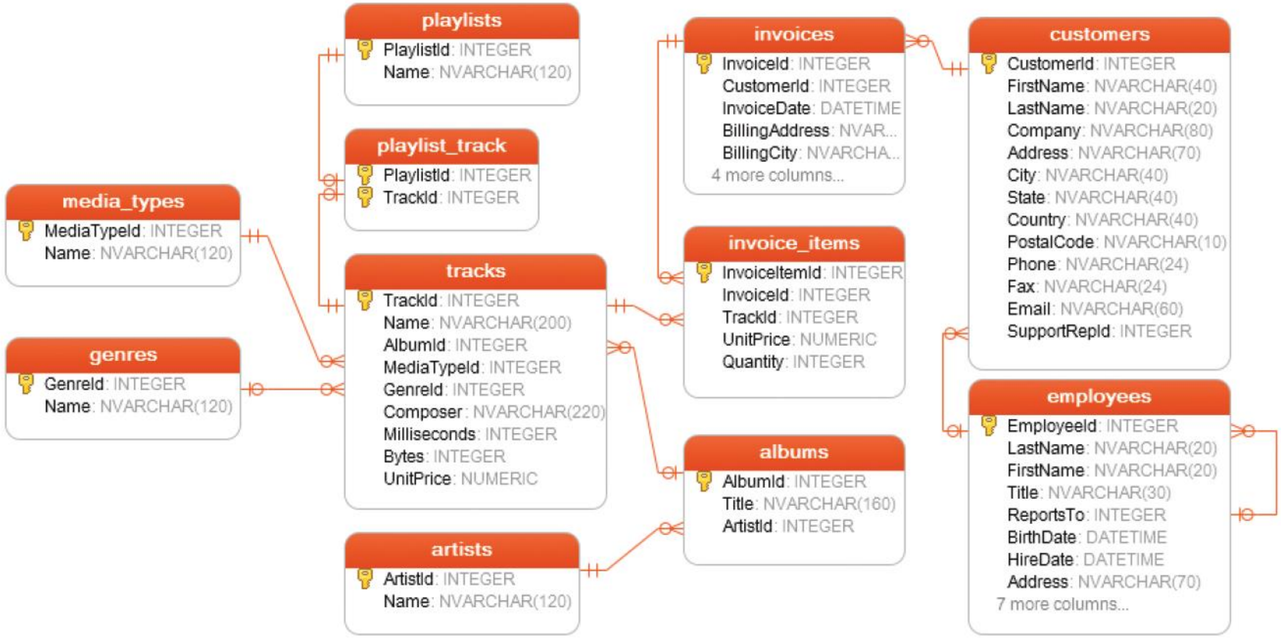
Write a query that returns track name and its composer using tracks table.



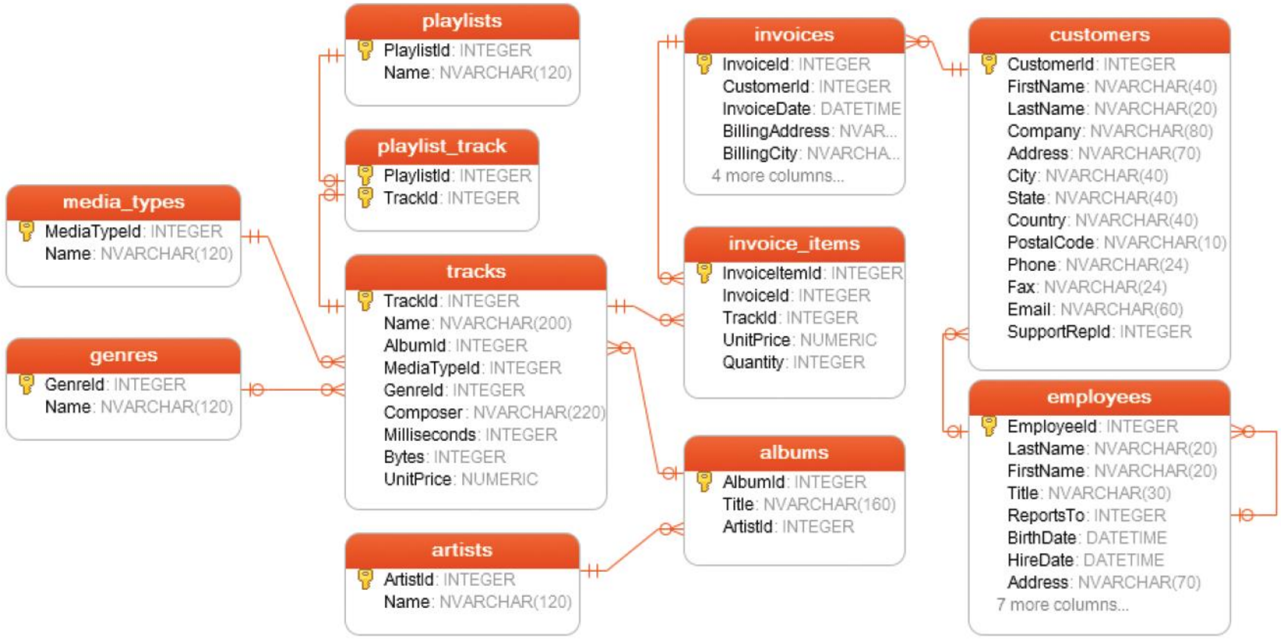
Write a query that returns all columns of albums table.



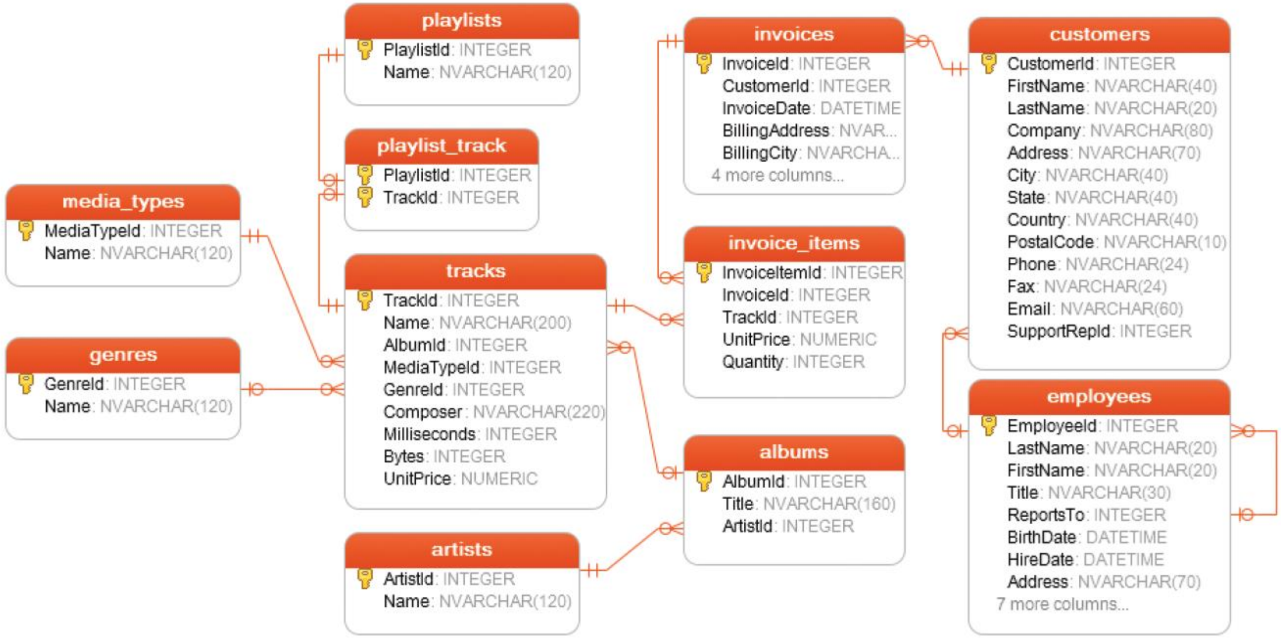
Write a query that returns columns of tracks table.



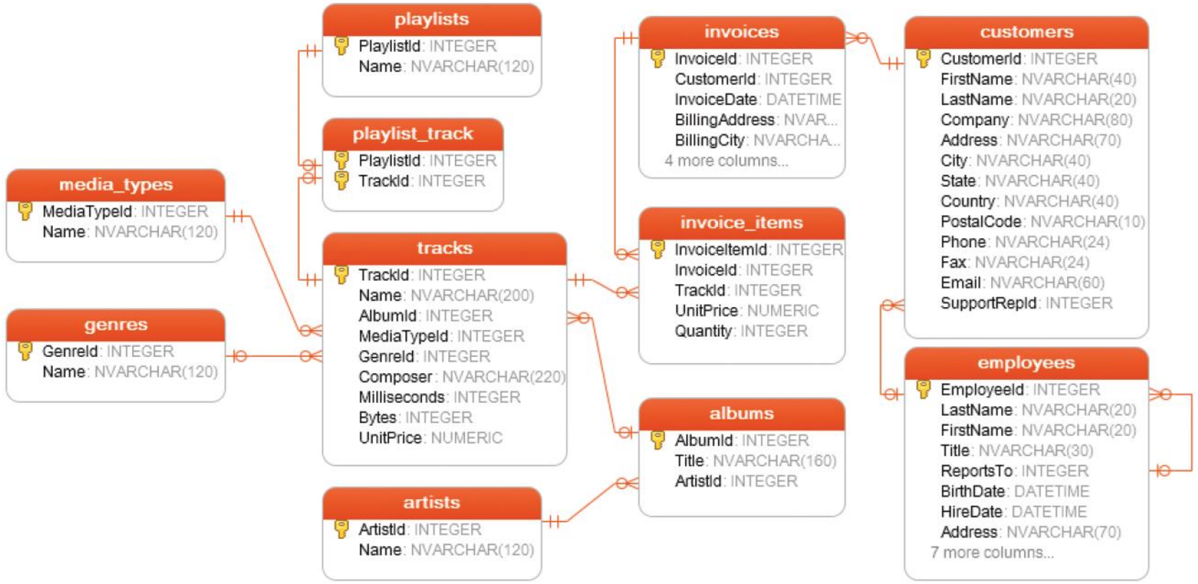
Find the name of composers of each track using tracks table.



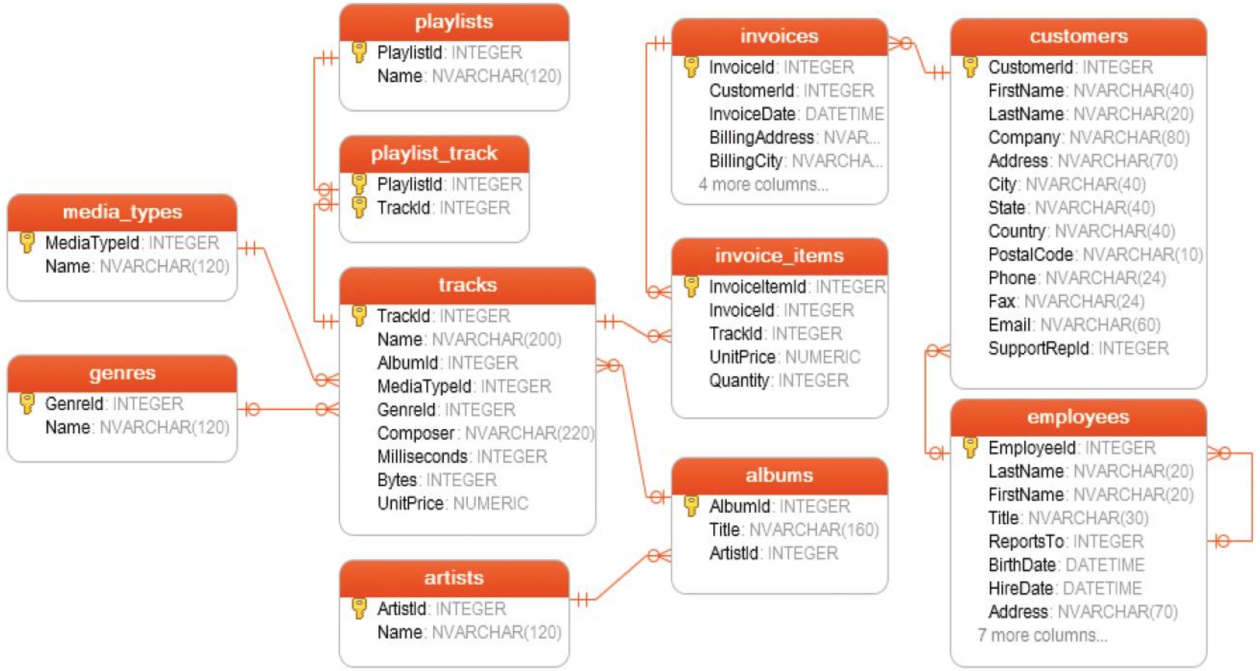
Write a query that return distinct AlbumId, MediaTypeId pair



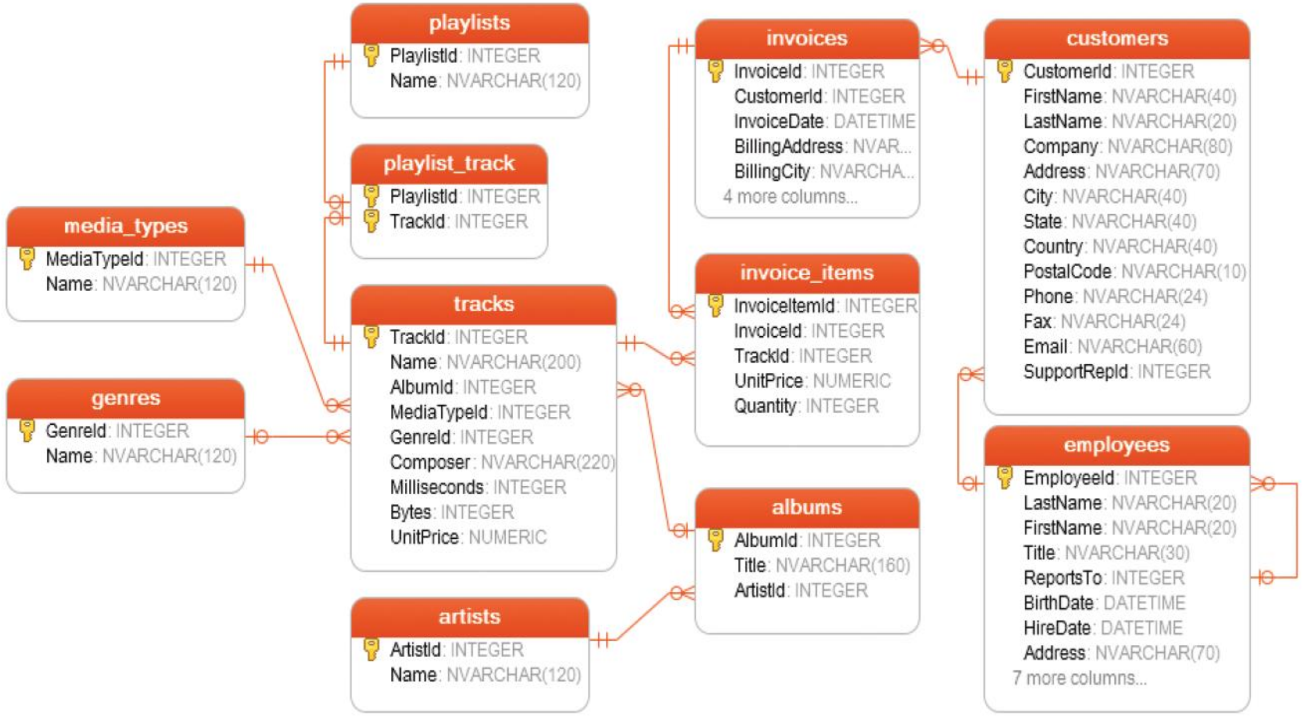
Find the track names of Jimi Hendrix.



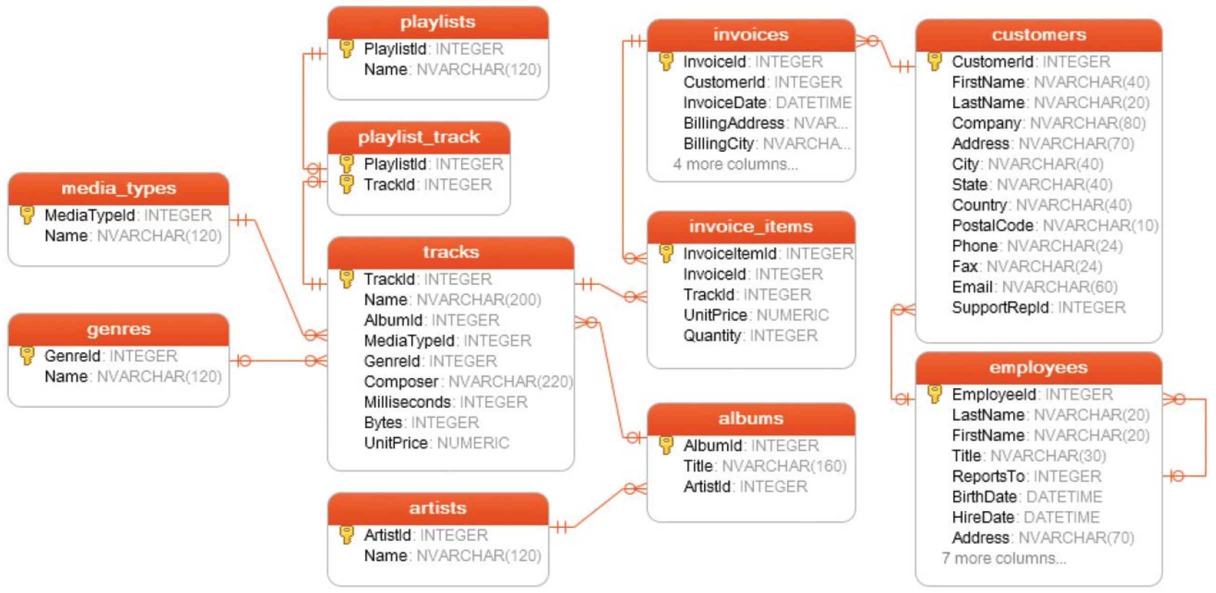
Find all the info of the invoices of which total amount is greater than $10.



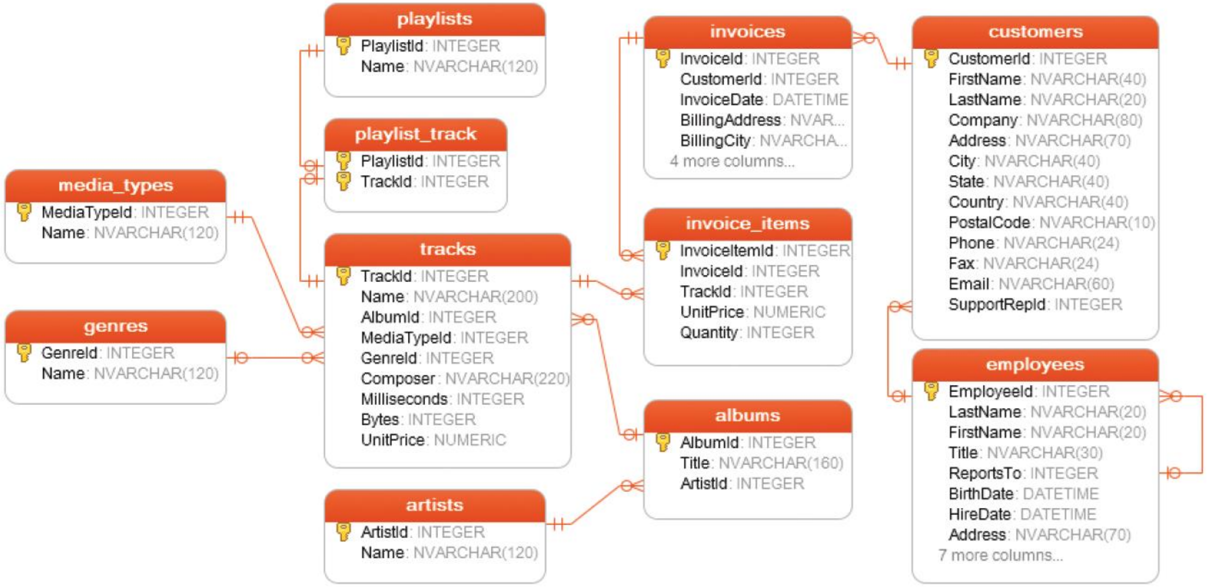
Find all the info of the invoices of which total amount is greater than $10. Just return the first 4



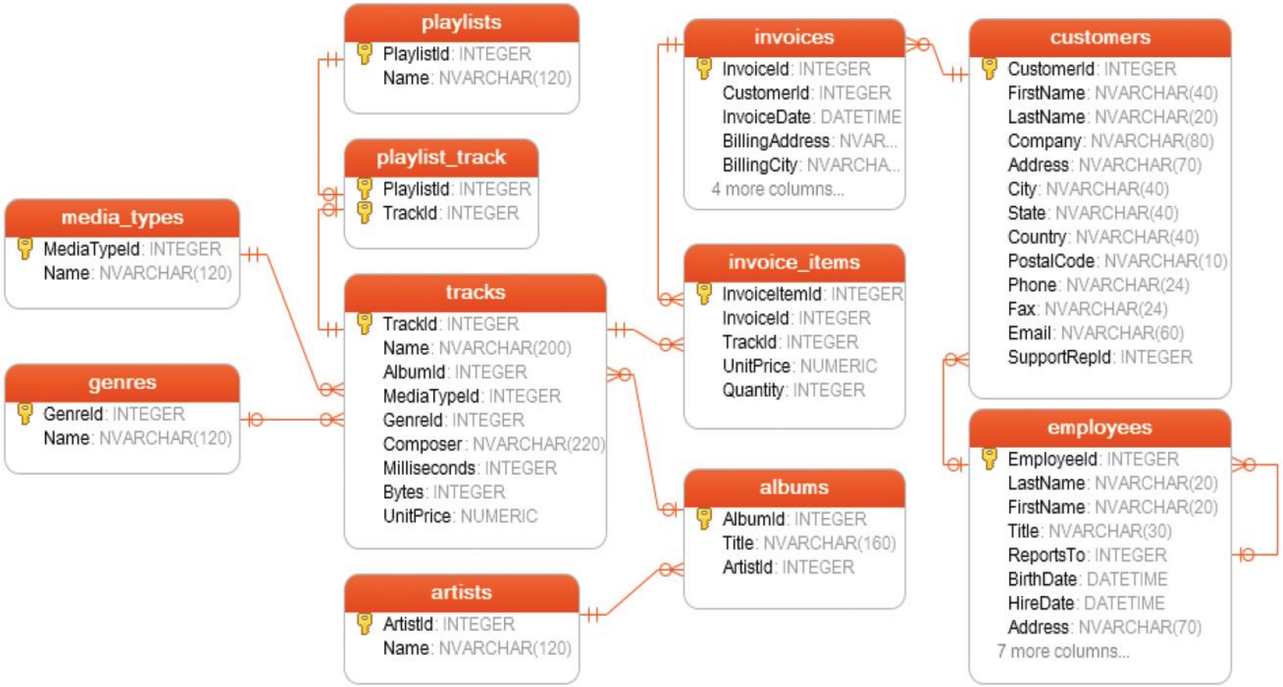
Find all the info of the invoices of which total amount is greater than $10. Then sort them by the total amount in descending order.



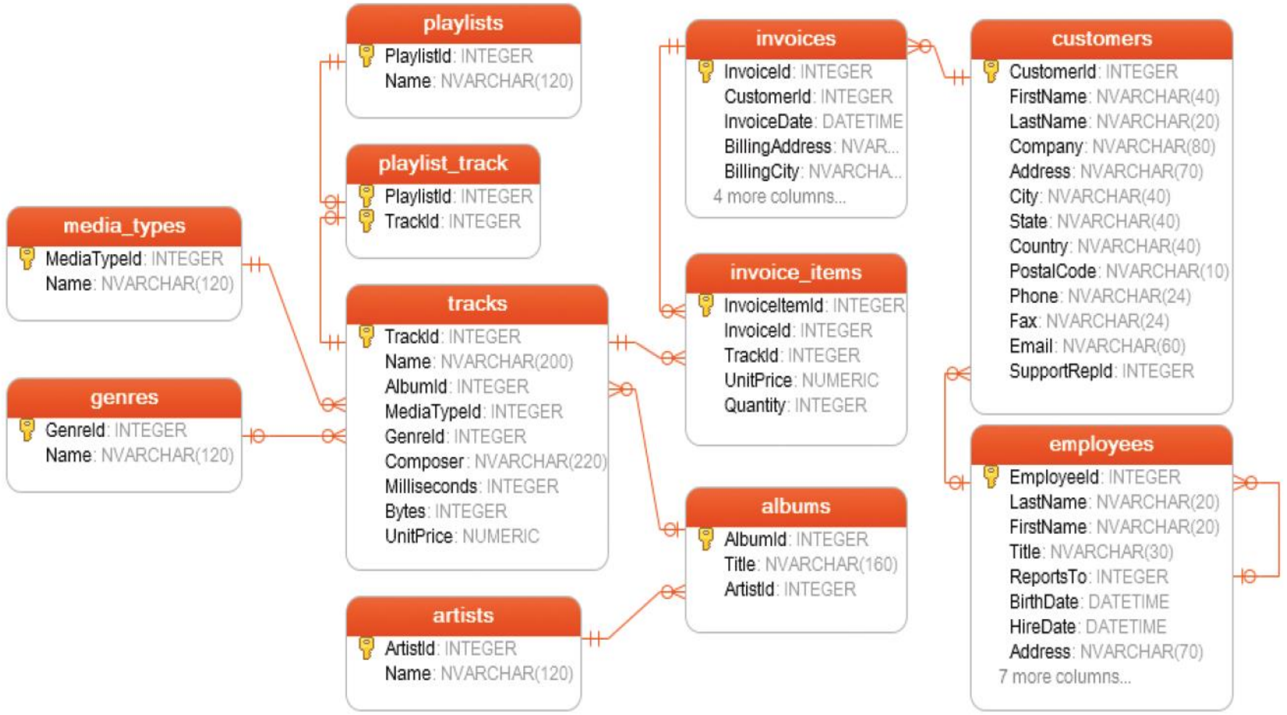
Find all the info of the invoices of which billing country is not USA. Then sort them by the total amount in ascending order.



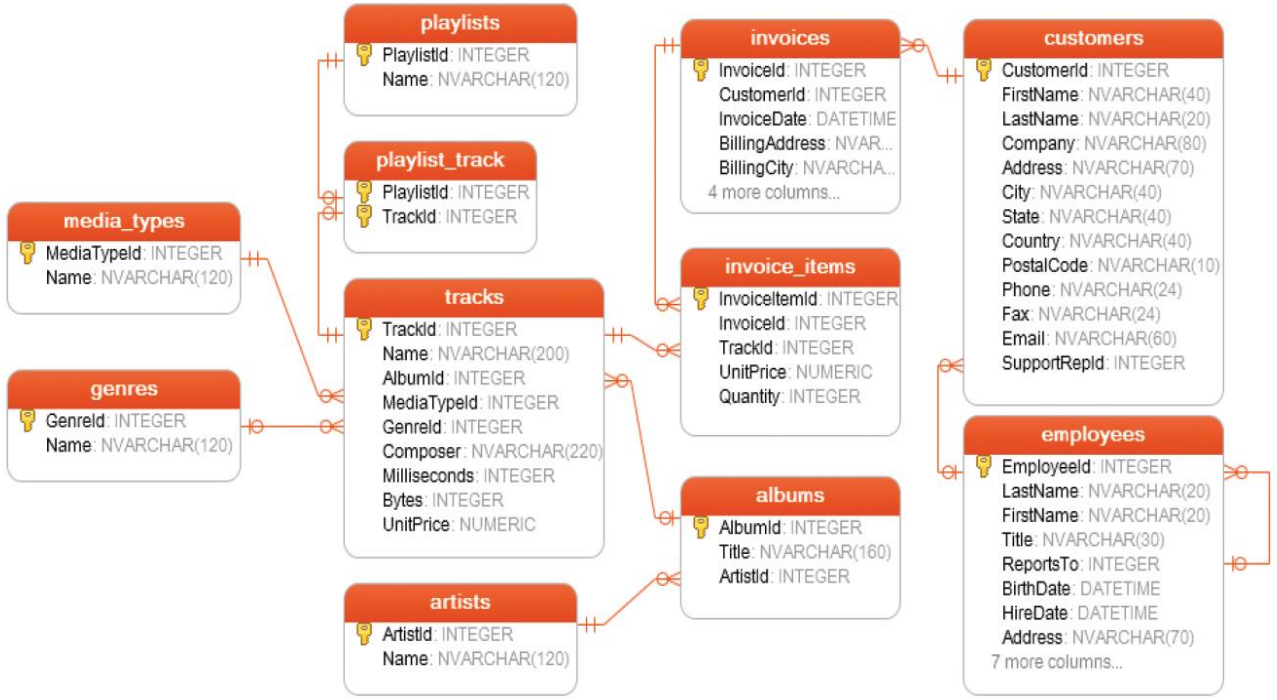
Find the newest invoice date among the invoice dates between 2009 and 2011.



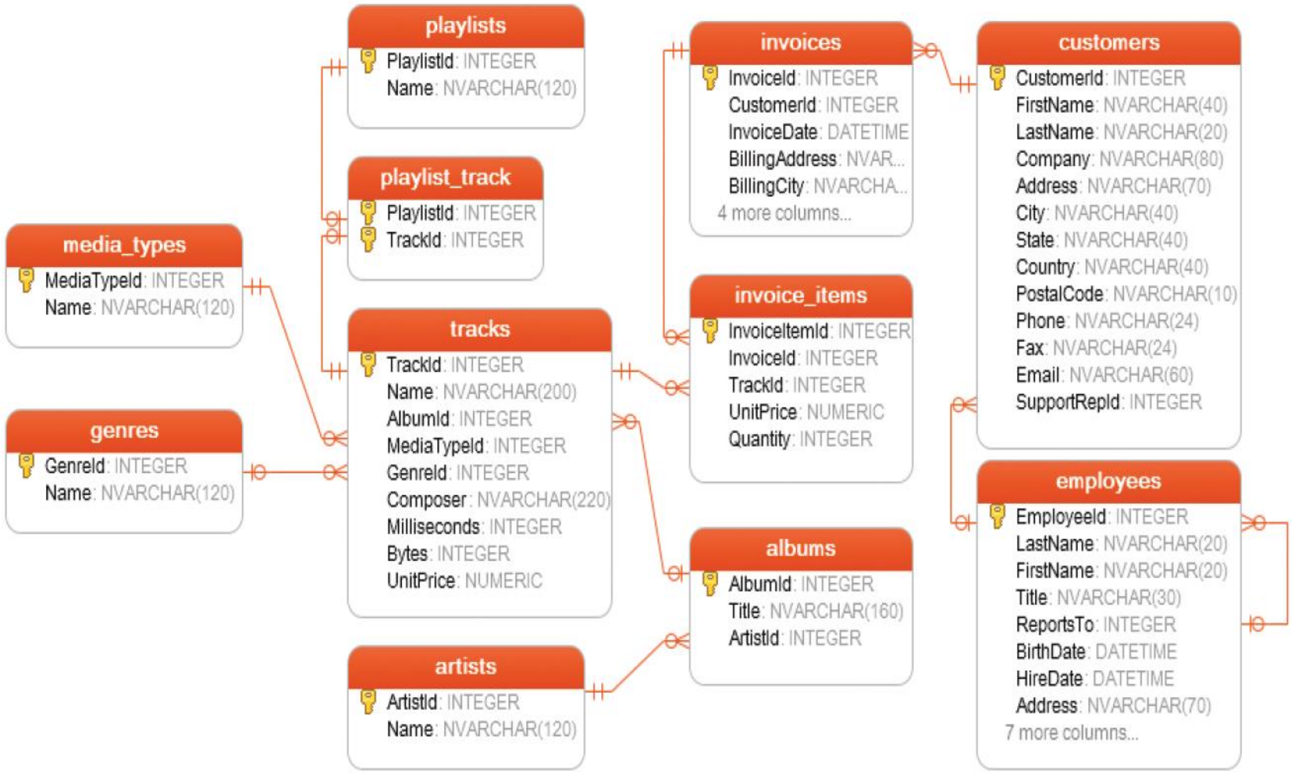
Find the first and last name of the customers who gave an order from Belgium, Norway, Canada and USA.



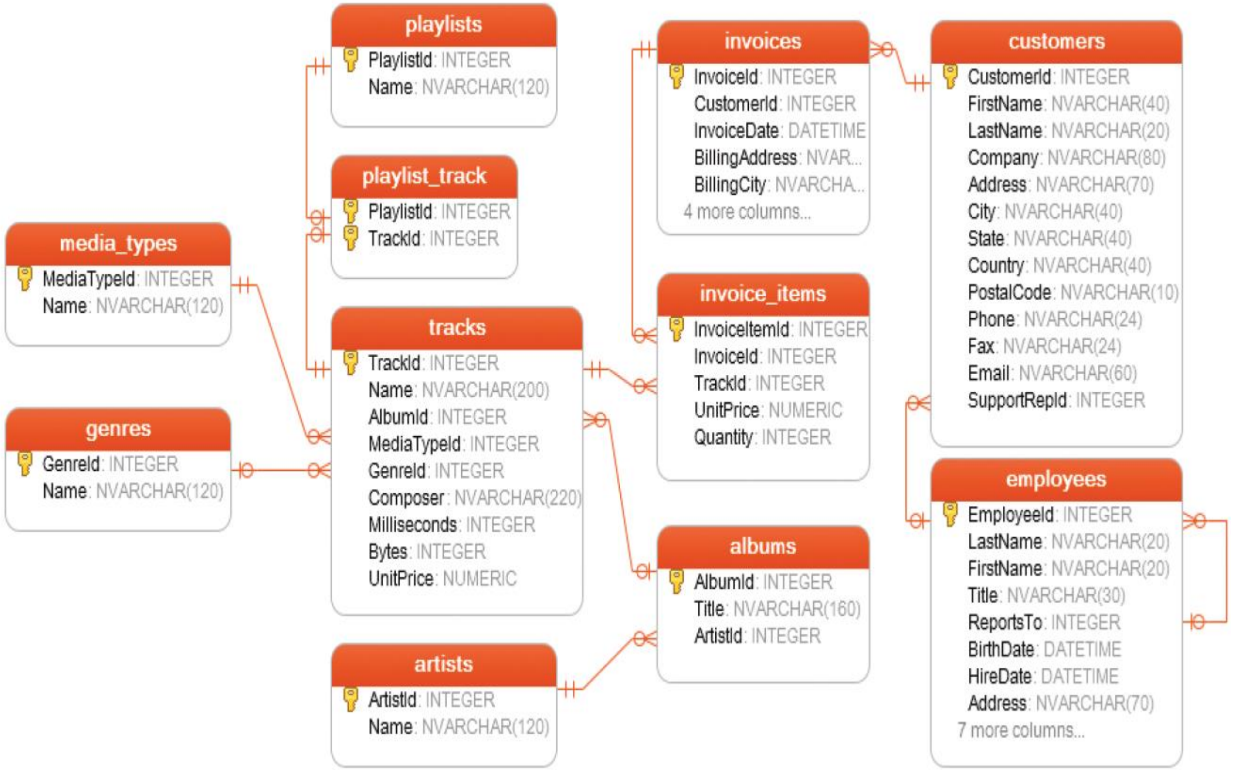
Find the track names of Bach.



How many invoices are in the digital music store?

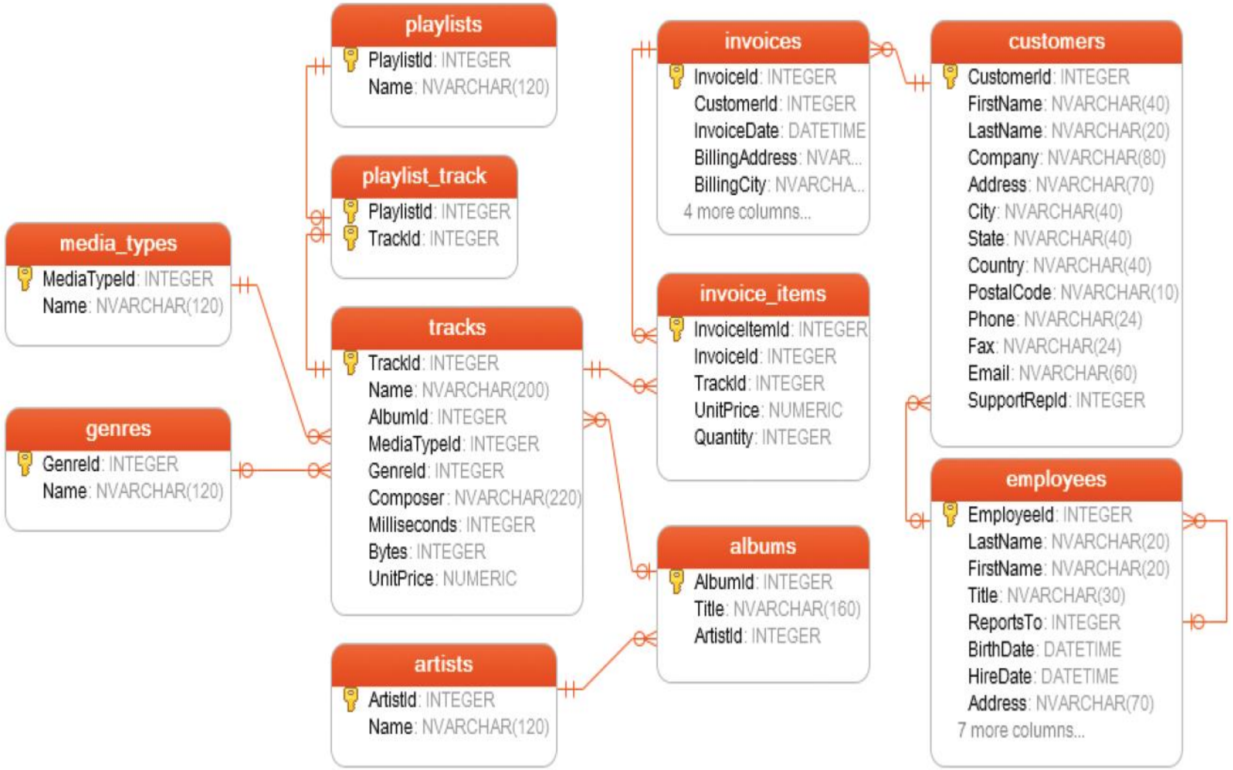


How many composers are there in the digital music store?



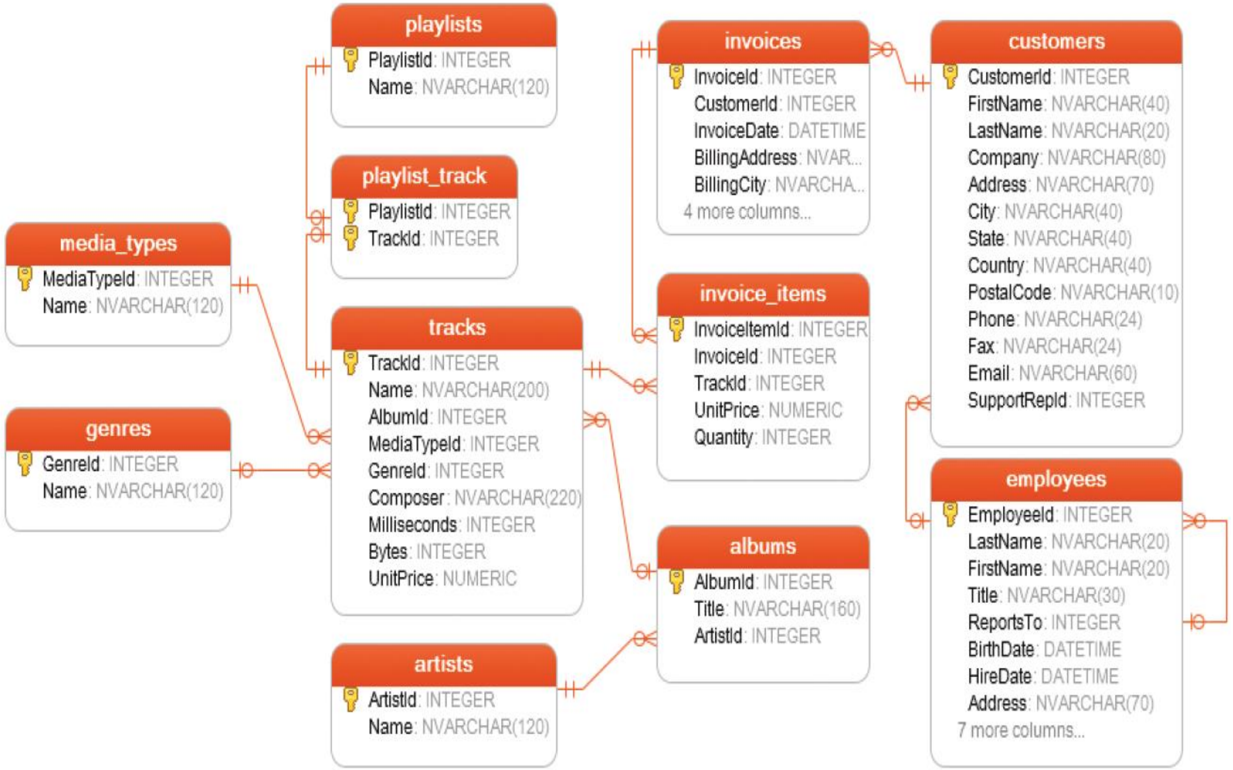
CLARUSWAY
WAY TO REINVENT YOURSELF

Find the track name having the minimum duration.



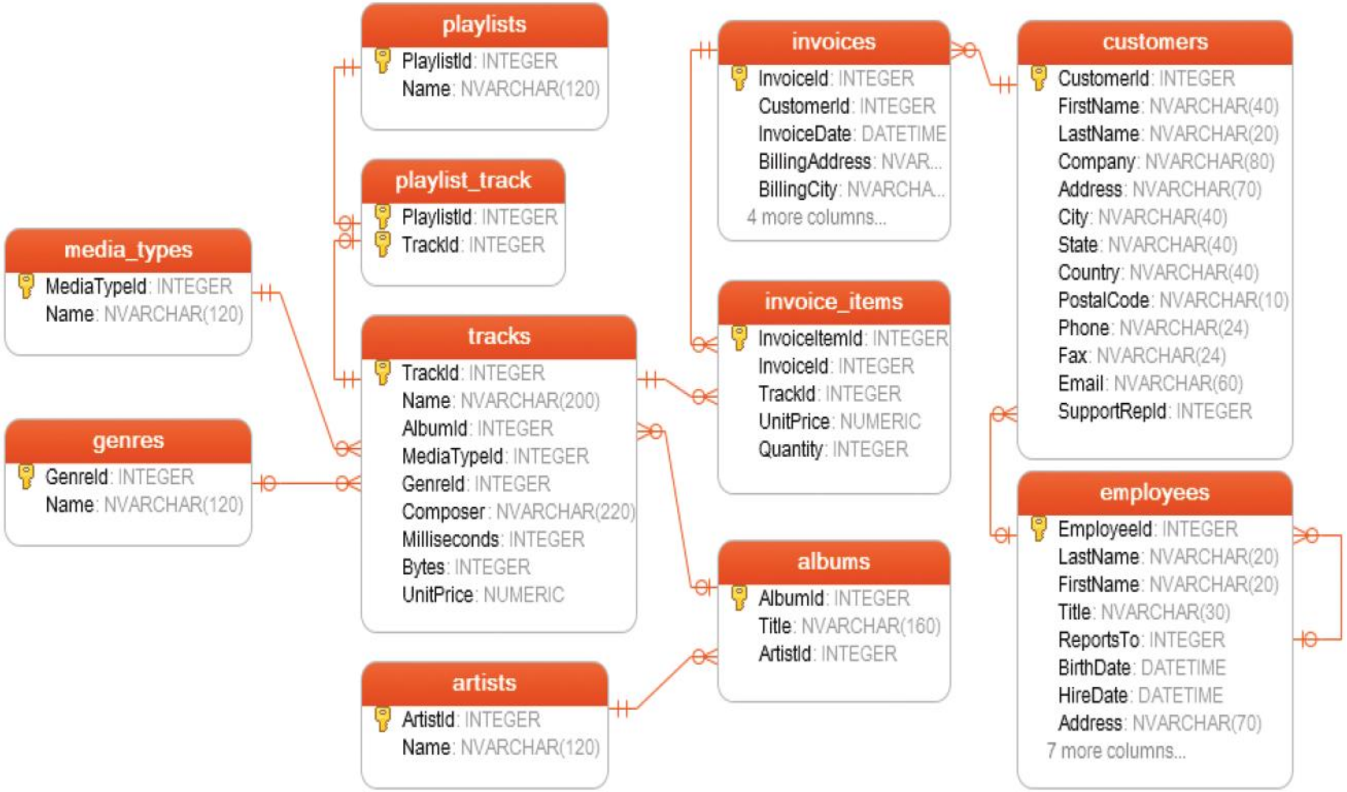
CLARUSWAY
WAY TO REINVENT YOURSELF

Find the track name having the maximum duration.

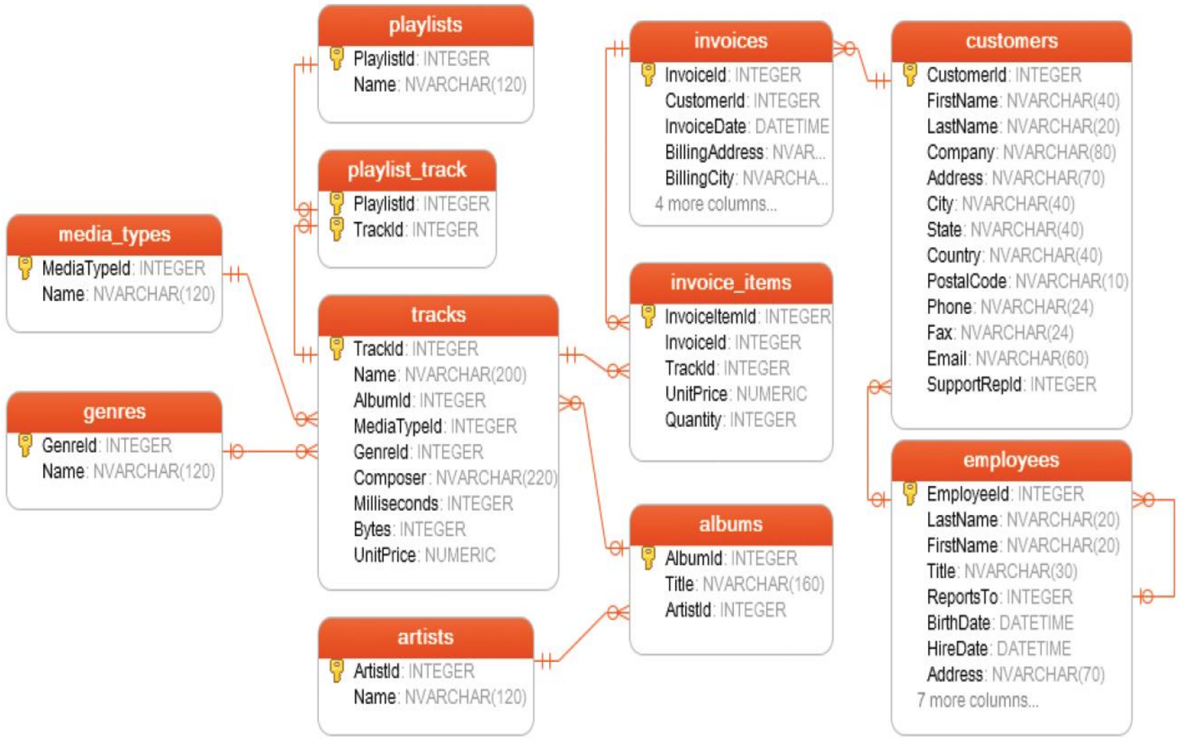


CLARUSWAY
WAY TO REINVENT YOURSELF

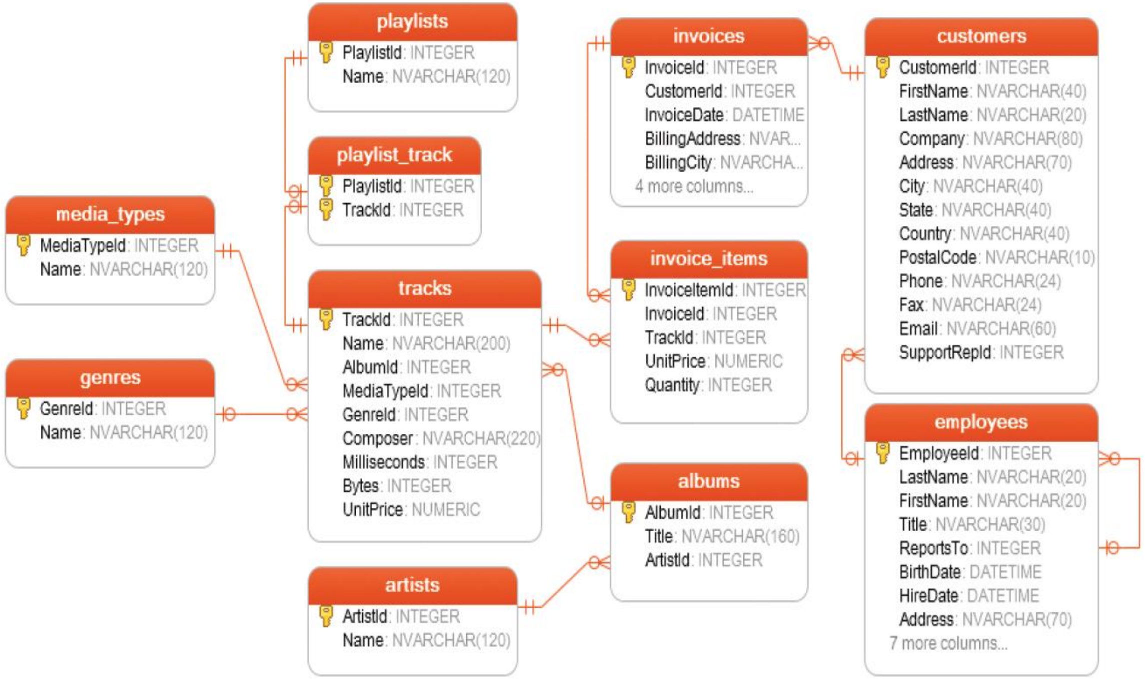
How much money did our store earn?



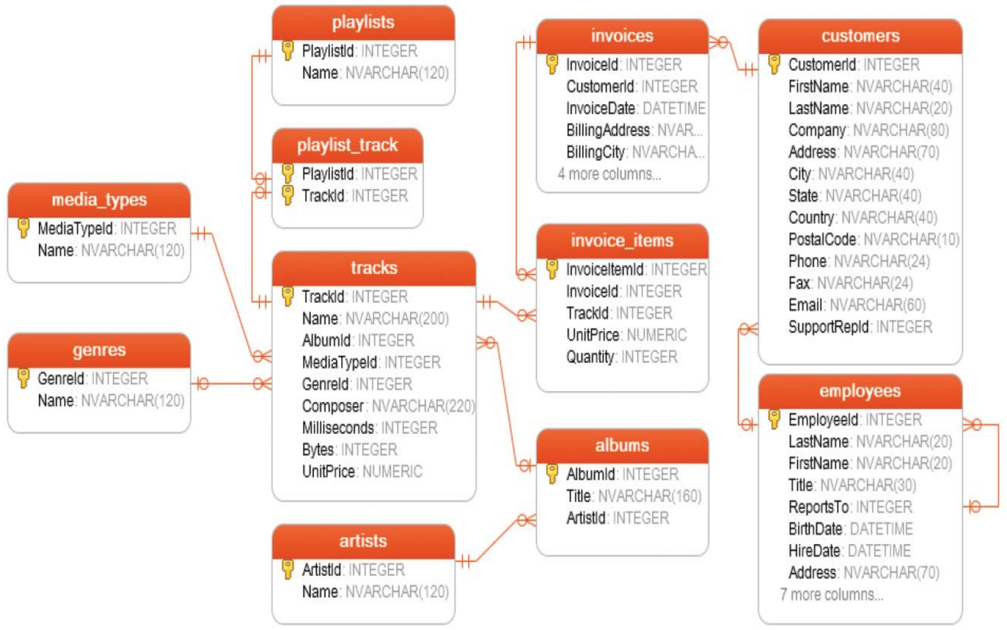
Find the tracks having duration bigger than the average duration.



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

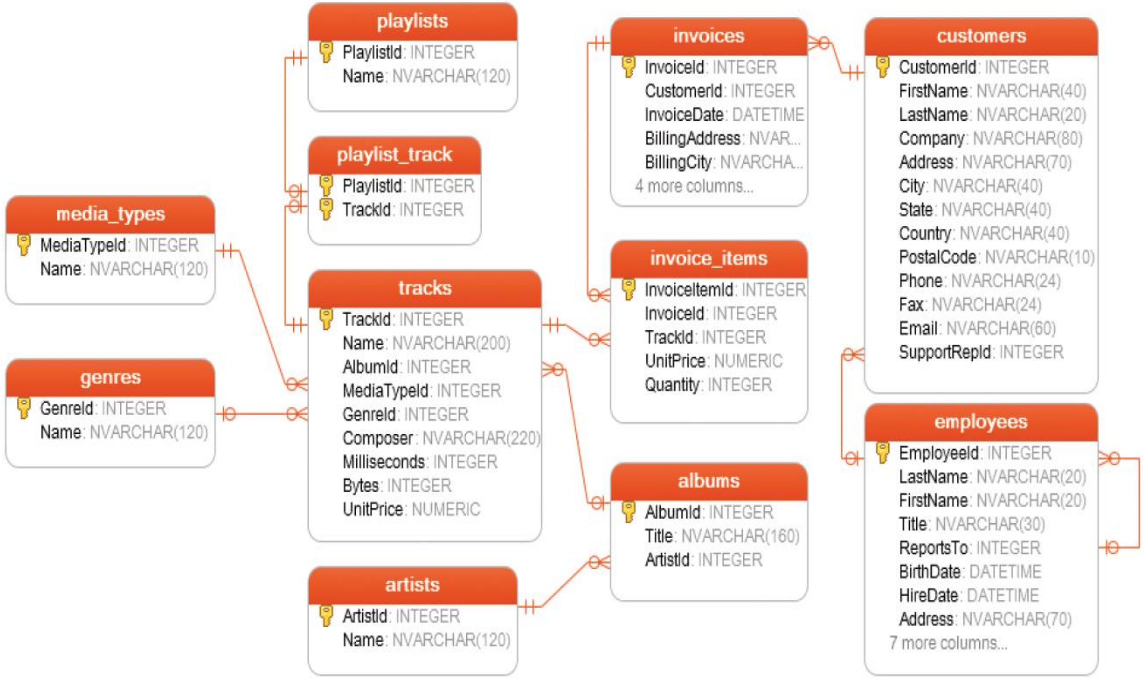


How many customers do we have from each country? Your result will include name of the country and number.

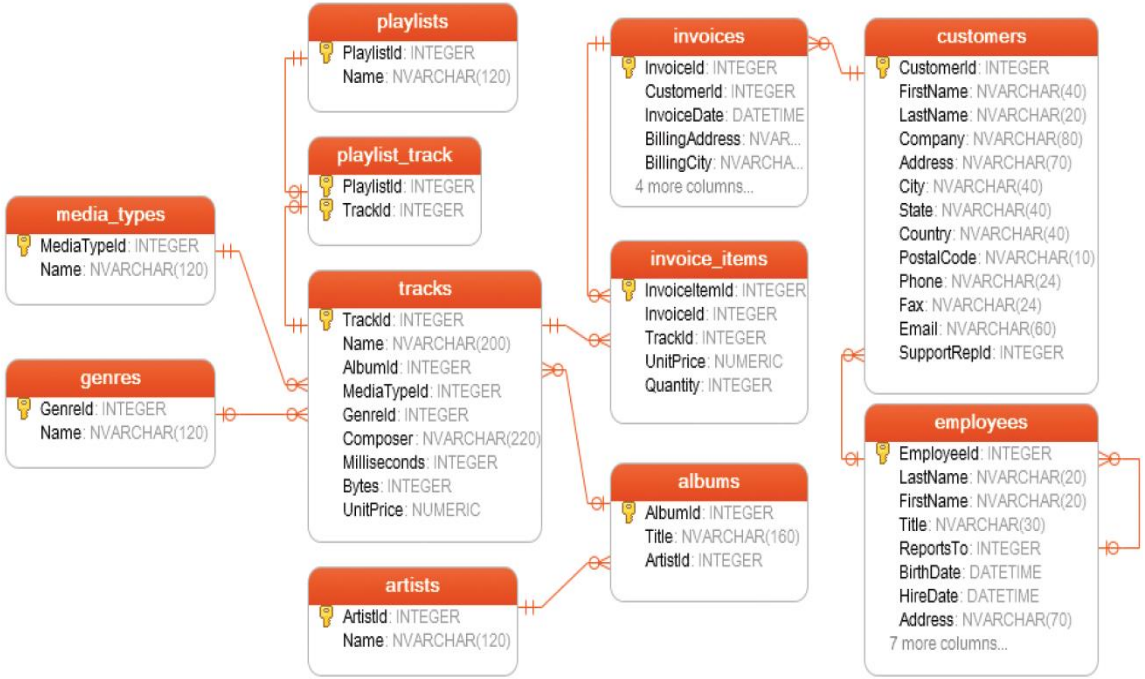


CLARUSWAY©
WAY TO REINVENT YOURSELF

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



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



Drag your dot to how you are feeling:

Keep going, I understand

I'm a little confused

Stop, I need help!

Students, drag the icon! ●

THANKS!

Any questions?