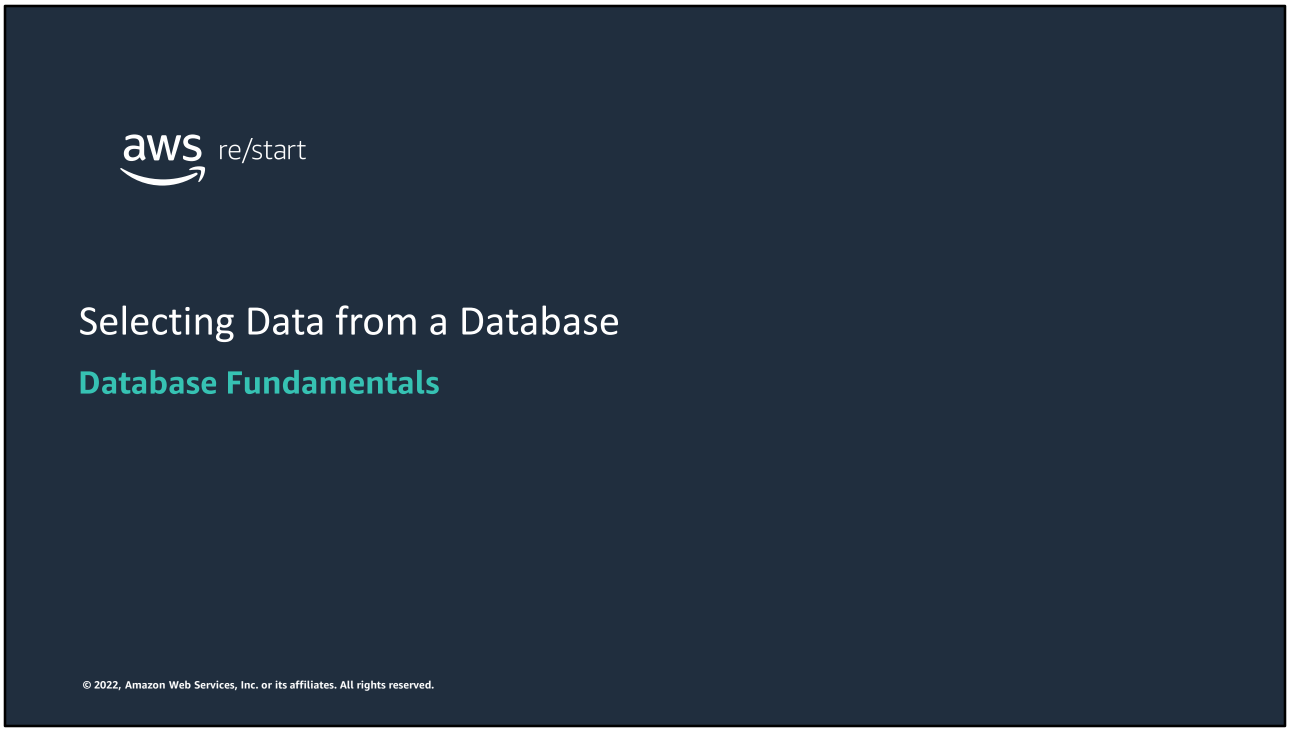
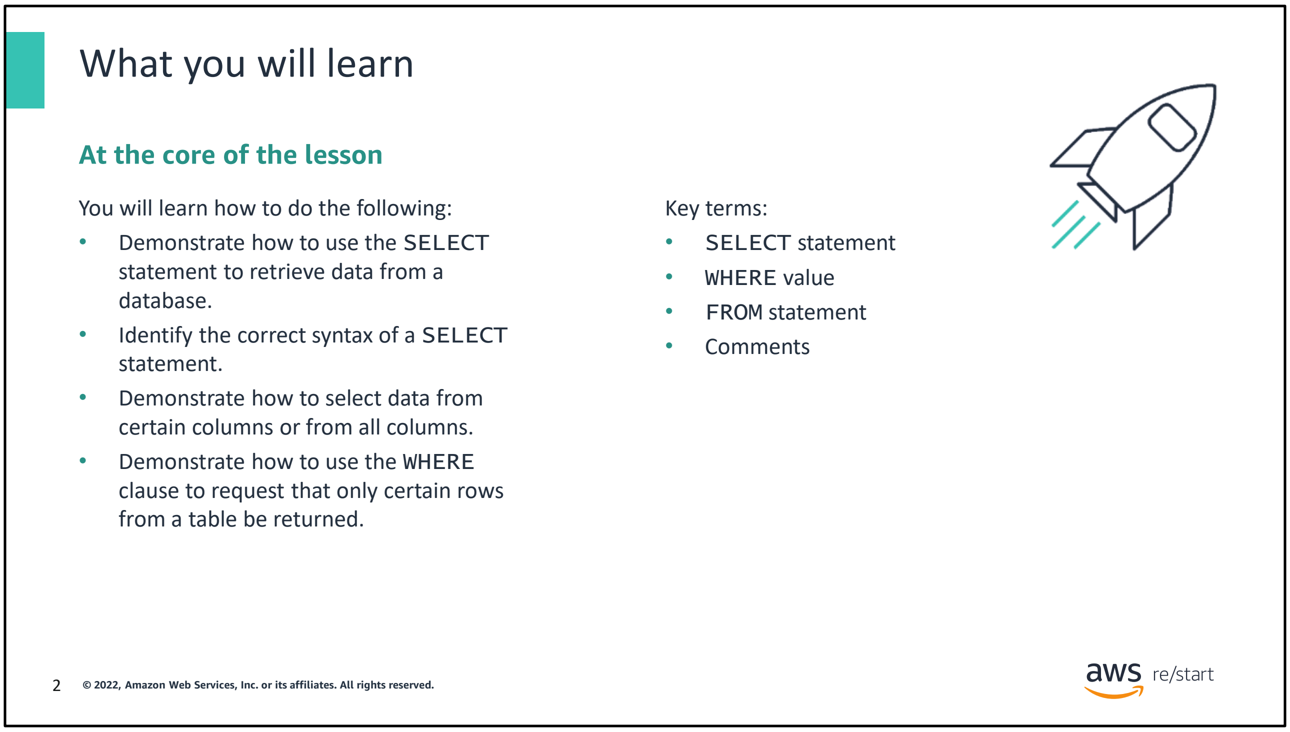


aws re/start

Selecting Data from a Database

Database Fundamentals

Welcome to Selecting Data from a Database.



What you will learn

At the core of the lesson

You will learn how to do the following:

- Demonstrate how to use the SELECT statement to retrieve data from a database.
- Identify the correct syntax of a SELECT statement.
- Demonstrate how to select data from certain columns or from all columns.
- Demonstrate how to use the WHERE clause to request that only certain rows from a table be returned.

Key terms:

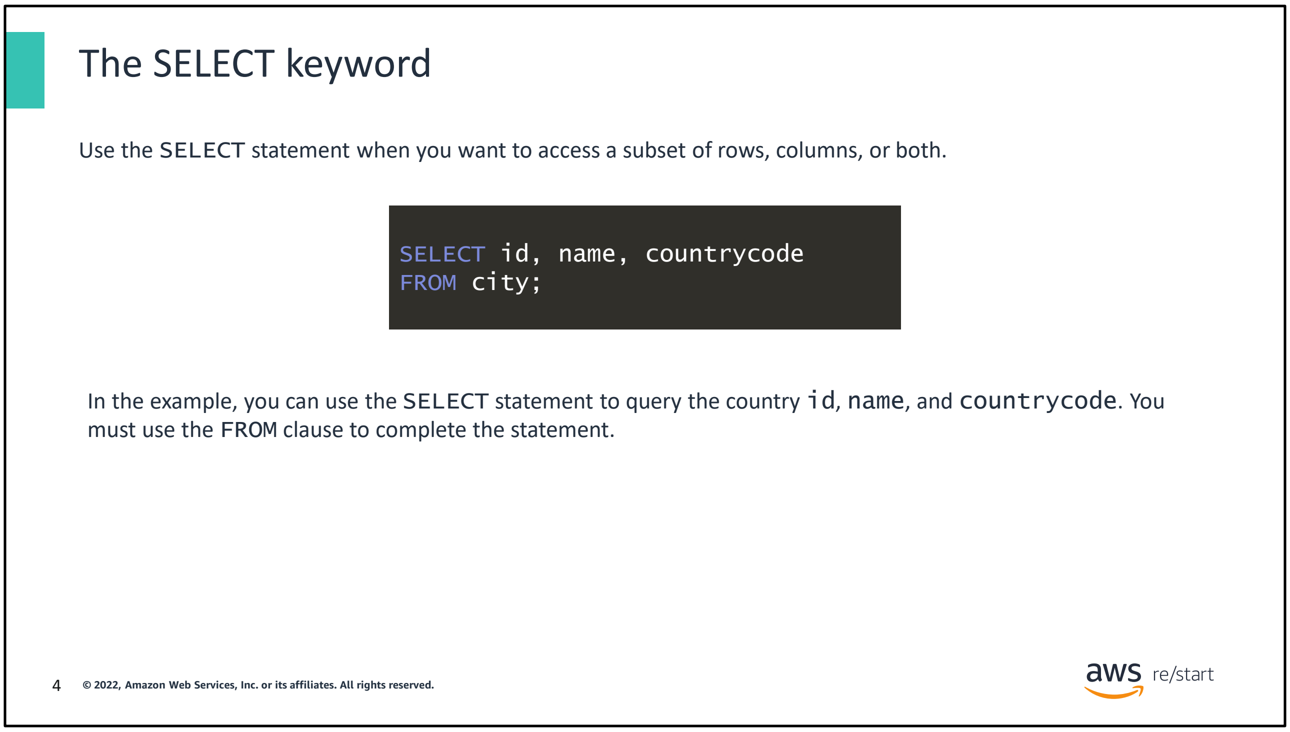
- SELECT statement
- WHERE value
- FROM statement
- Comments

In this module, you will learn how to do the following:

- Demonstrate how to use the SELECT statement to retrieve data from a database.
- Identify the correct syntax of a SELECT statement.
- Demonstrate how to select data from certain columns or from all columns.
- Demonstrate how to use the WHERE clause to request that only certain rows from a table be returned.

SELECT statement

When selecting data from a database, you use the SELECT statement.



The SELECT keyword

Use the SELECT statement when you want to access a subset of rows, columns, or both.

```
SELECT id, name, countrycode
FROM city;
```

In the example, you can use the SELECT statement to query the country id, name, and countrycode. You must use the FROM clause to complete the statement.

You use the SELECT statement to select one or more columns from a table. You can also use the SELECT statement when you want to access a subset of rows, columns, or both. When you query tables, you must include the FROM clause in your syntax. The result of the SELECT statement is called a *result set*. It lists rows that contain the same number of columns.

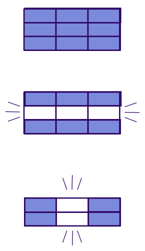
As you read from left to right, the statement begins with SELECT. Next, you see which columns should be returned and from which table. The FROM clause follows immediately after the SELECT portion.

How it works

When thinking about how a query is processed, it is important to remember that the query is processed out of order. The query will pull all of the data from the specified table and then work through each of the clauses as the example shows.

Query

```
SELECT id, name, countrycode
FROM city
WHERE countrycode ='BRA';
```



Statement order of operation:

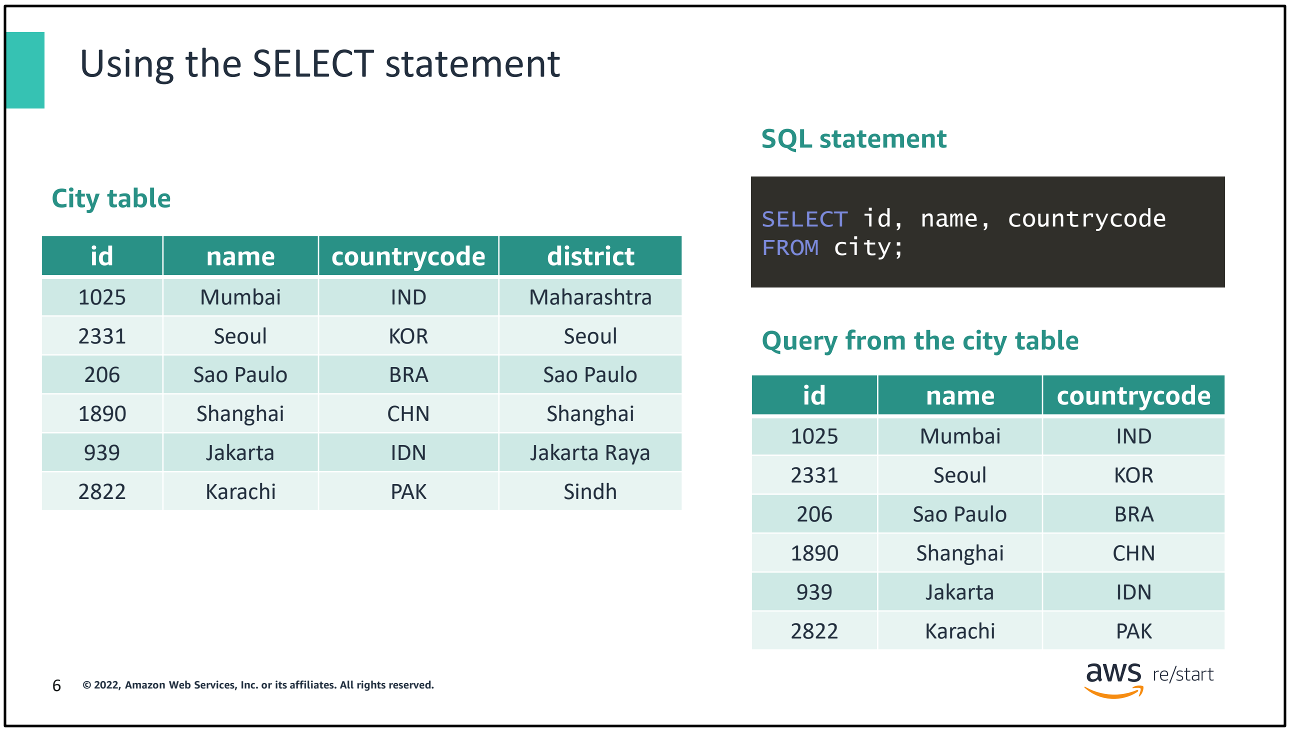
1. FROM the city table, get all data.
2. WHERE the countrycode is BRA, keep the row. (Ignore the others.)
3. SELECT the specified columns (`id`, `name`, and `countrycode`), and ignore the others.

The following query is for this example:

```
SELECT id, name, countrycode,
FROM city
WHERE countrycode ='BRA’;
```

However, the order in which the query is processed is as follows:

1. FROM the city table, get all data.
2. WHERE the countrycode is BRA, keep the row, and ignore the others.
3. SELECT the specified columns (`id`, `name`, and `countrycode`), and ignore the others.



Using the SELECT statement

City table

id	name	countrycode	district
1025	Mumbai	IND	Maharashtra
2331	Seoul	KOR	Seoul
206	Sao Paulo	BRA	Sao Paulo
1890	Shanghai	CHN	Shanghai
939	Jakarta	IDN	Jakarta Raya
2822	Karachi	PAK	Sindh

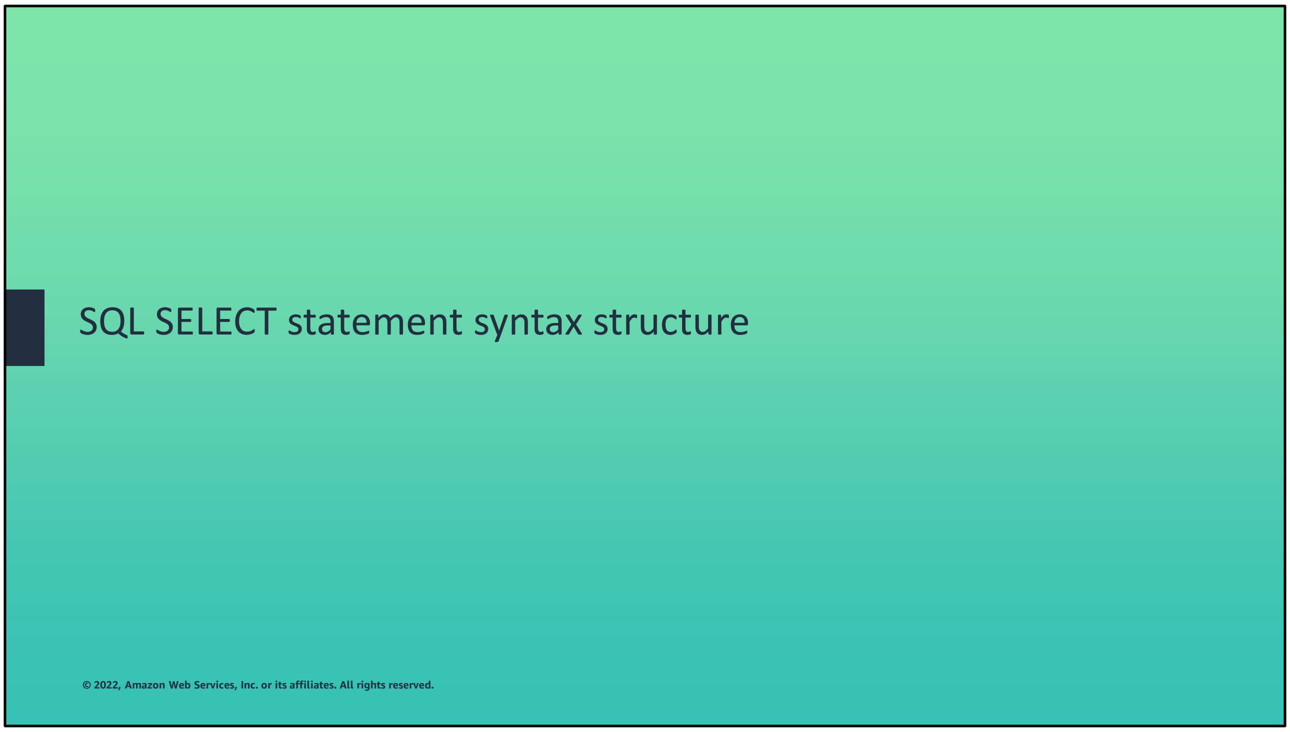
SQL statement

```
SELECT id, name, countrycode
FROM city;
```

Query from the city table

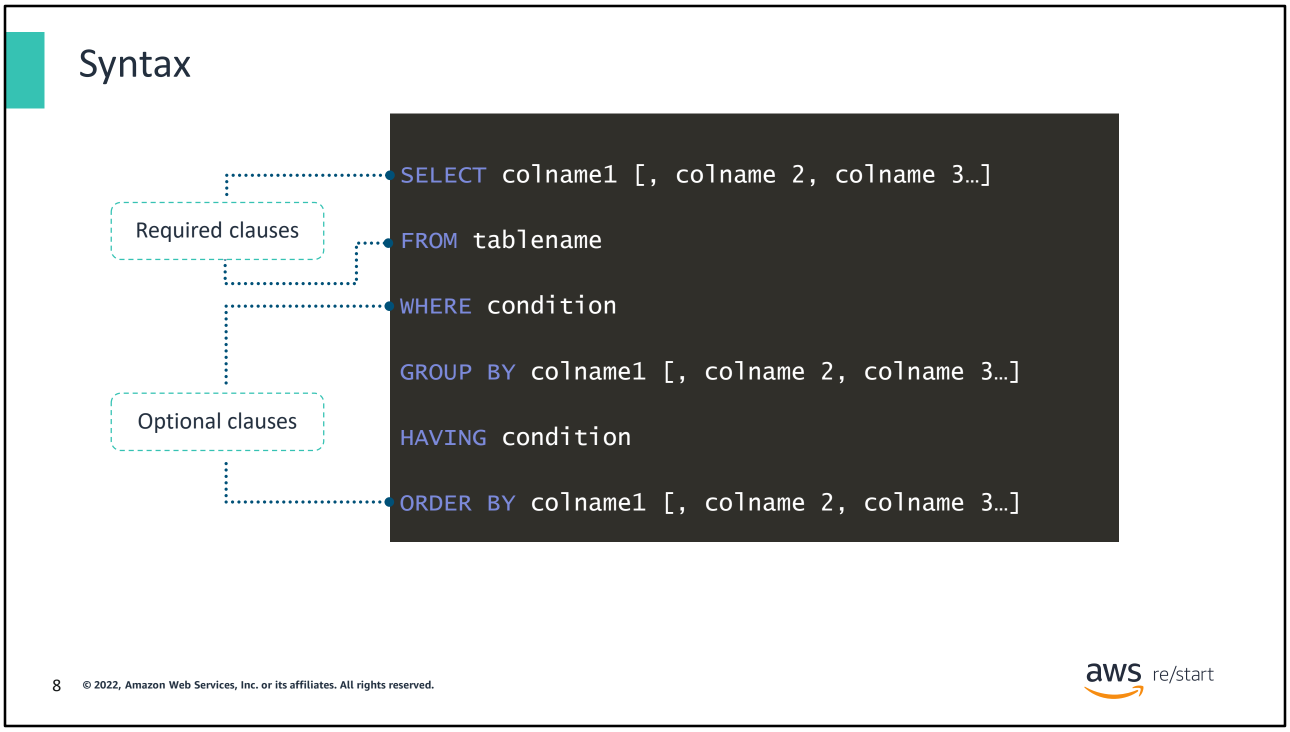
id	name	countrycode
1025	Mumbai	IND
2331	Seoul	KOR
206	Sao Paulo	BRA
1890	Shanghai	CHN
939	Jakarta	IDN
2822	Karachi	PAK

The original table includes the following: `id, name, countrycode,` and `district`. By using the SELECT statement, the queried table shows the data limited to the columns requested in the query (`id, name,` and `countrycode`).

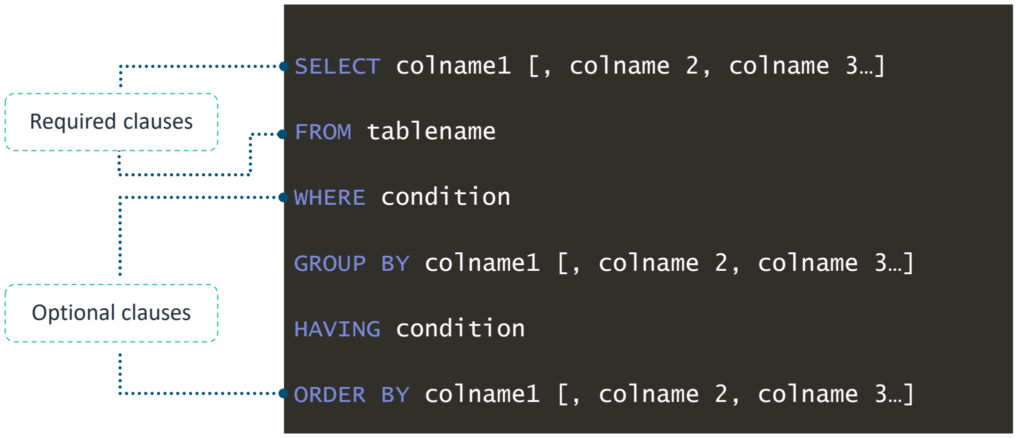


SQL SELECT statement syntax structure

When using the SELECT statement, it is important to remember the syntax structure of the statement.



Syntax

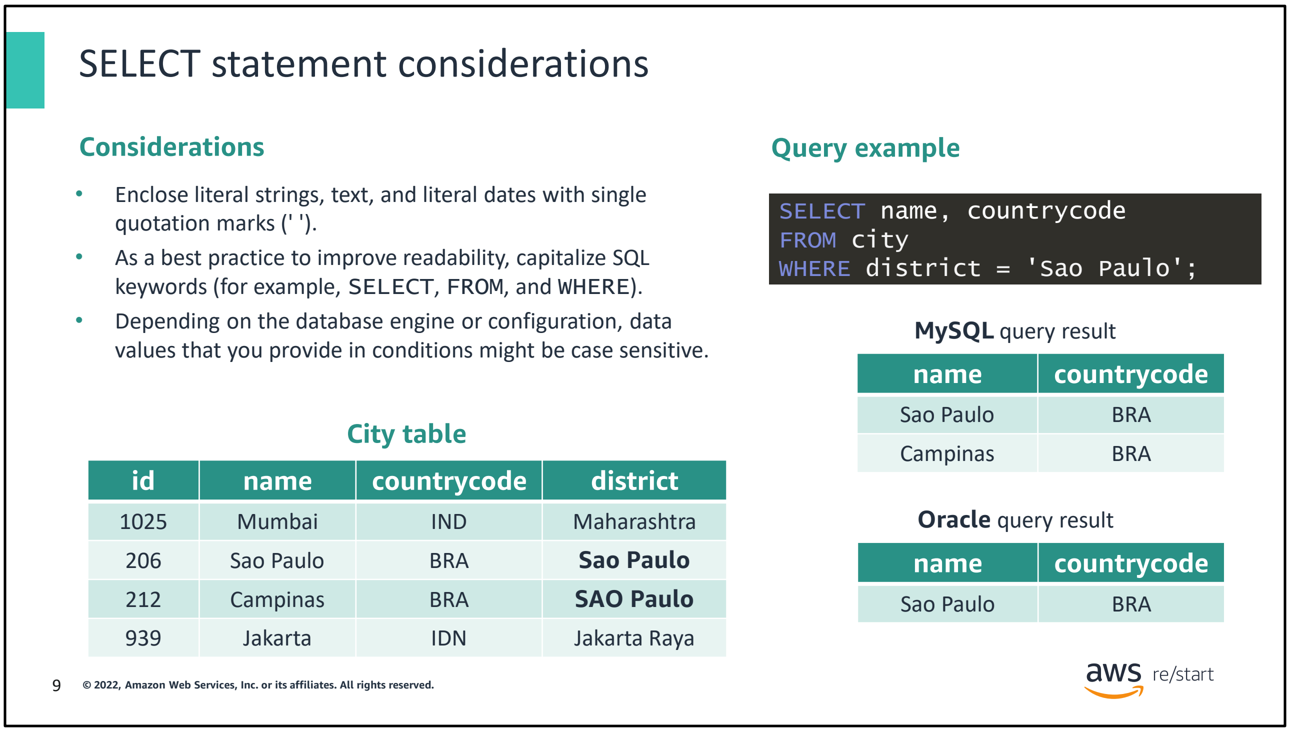


```
SELECT colname1 [, colname 2, colname 3…]
FROM tablename
WHERE condition
GROUP BY colname1 [, colname 2, colname 3…]
HAVING condition
ORDER BY colname1 [, colname 2, colname 3…]
```

The syntax for selecting data follows a precise order. The required clauses must precede the optional clauses.

The first clause contains SELECT and the column names, and the FROM clause with the table name immediately follows it.

All optional clauses will follow these first two required clauses.



SELECT statement considerations

Considerations

- Enclose literal strings, text, and literal dates with single quotation marks (' ').
- As a best practice to improve readability, capitalize SQL keywords (for example, `SELECT`, `FROM`, and `WHERE`).
- Depending on the database engine or configuration, data values that you provide in conditions might be case sensitive.

City table

id	name	countrycode	district
1025	Mumbai	IND	Maharashtra
206	Sao Paulo	BRA	**Sao Paulo**
212	Campinas	BRA	**SAO Paulo**
939	Jakarta	IDN	Jakarta Raya

Query example

```
SELECT name, countrycode
FROM city
WHERE district = 'Sao Paulo';
```

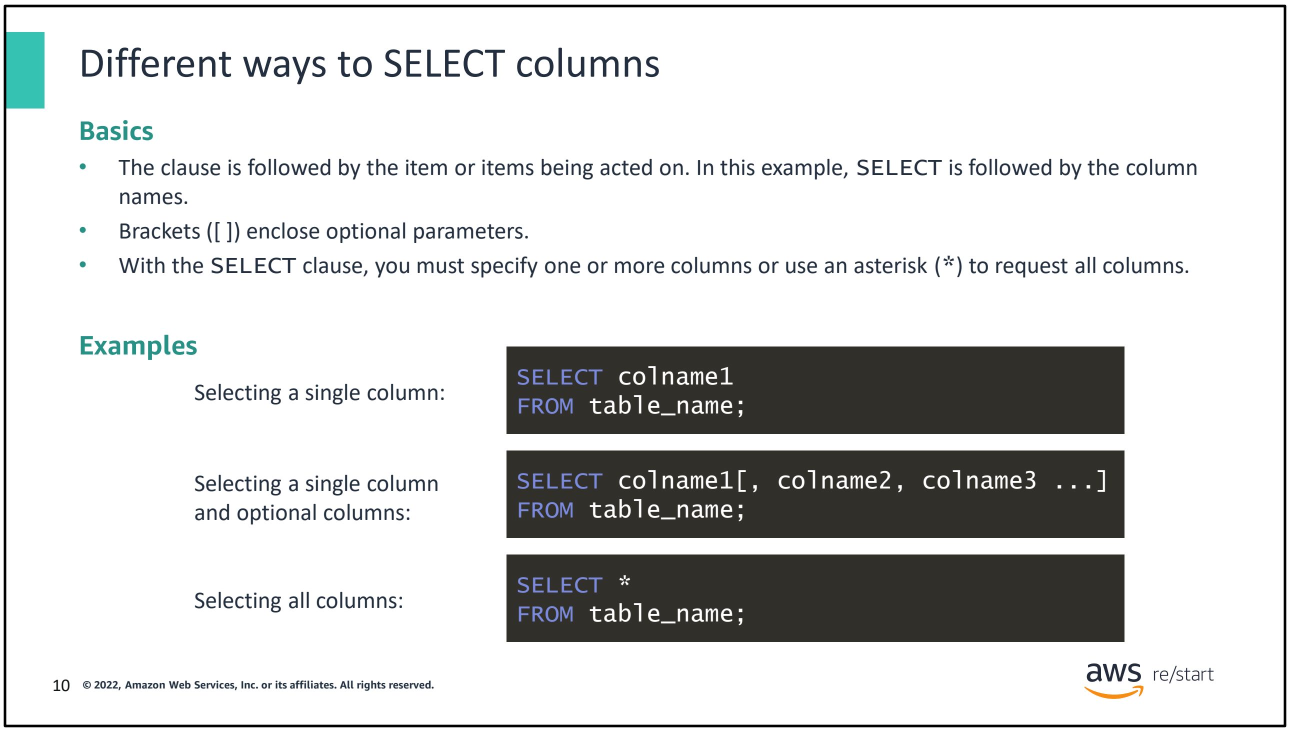
MySQL query result

name	countrycode
Sao Paulo	BRA
Campinas	BRA

Oracle query result

name	countrycode
Sao Paulo	BRA

This slide lists some considerations when using the SQL SELECT statement. In particular, data values that you provide in conditions might be case sensitive depending on the SQL database engine or configuration that you are using. In the example that's shown, the same query returns two different results depending on whether your database is MySQL or Oracle. MySQL is not case sensitive, but Oracle is.



Different ways to SELECT columns

Basics

- The clause is followed by the item or items being acted on. In this example, SELECT is followed by the column names.
- Brackets ([ ]) enclose optional parameters.
- With the SELECT clause, you must specify one or more columns or use an asterisk (*) to request all columns.

Examples

Selecting a single column:

```
SELECT colname1
FROM table_name;
```

Selecting a single column and optional columns:

```
SELECT colname1[, colname2, colname3 ...]
FROM table_name;
```

Selecting all columns:

```
SELECT *
FROM table_name;
```

The following information is about syntax notation.

The following are the basics of syntax:
- The clause is followed by the specific items that are acted on. In this example, the column names follow the SELECT clause.
- Brackets ([ ]) enclose optional parameters.
- With the SELECT clause, you must specify one or more columns or use an asterisk (*) to request all columns.

Here are some examples:
- Selecting a single column:

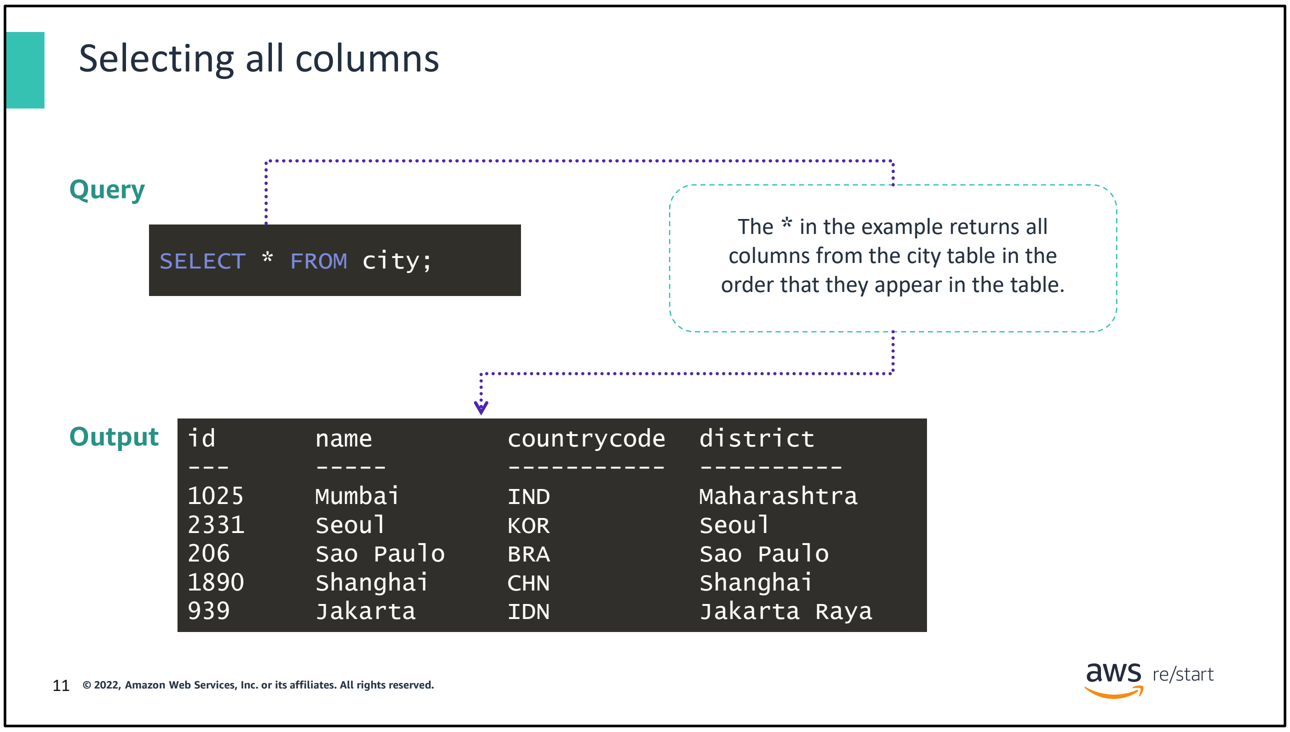
```
SELECT colname1
FROM table_name;
```

- Selecting a single column and optional columns:

```
SELECT colname1[, colname2, colname3 ...]
FROM table_name;
```

- Selecting all columns:

```
SELECT *
FROM table_name;
```



Selecting all columns

Query

```
SELECT * FROM city;
```

The * in the example returns all columns from the city table in the order that they appear in the table.

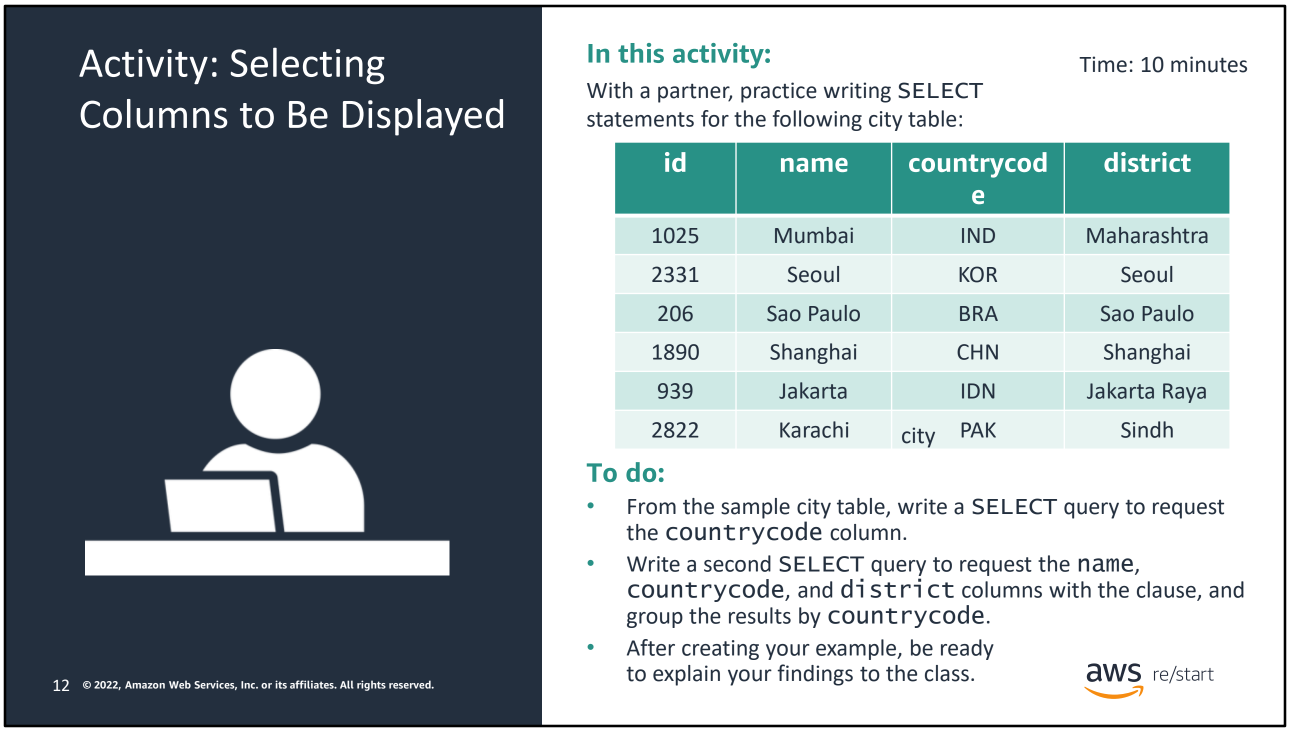
Output

```
id      name        countrycode  district
---     -----       -----------  ----------
1025    Mumbai      IND          Maharashtra
2331    Seoul       KOR          Seoul
206     Sao Paulo   BRA          Sao Paulo
1890    Shanghai    CHN          Shanghai
939     Jakarta     IDN          Jakarta Raya
```

The * in the example returns all columns from the city table in the order that they appear in the table.

- Syntax:

```
SELECT * FROM city;
```



Activity: Selecting Columns to Be Displayed

In this activity:

Time: 10 minutes

With a partner, practice writing SELECT statements for the following city table:

id	name	countrycode	district
1025	Mumbai	IND	Maharashtra
2331	Seoul	KOR	Seoul
206	Sao Paulo	BRA	Sao Paulo
1890	Shanghai	CHN	Shanghai
939	Jakarta	IDN	Jakarta Raya
2822	Karachi	city PAK	Sindh

To do:

- From the sample city table, write a SELECT query to request the `countrycode` column.
- Write a second SELECT query to request the `name`, `countrycode`, and `district` columns with the clause, and group the results by `countrycode`.
- After creating your example, be ready to explain your findings to the class.

aws re/start

With a partner, practice writing SELECT statements for the following city table:

To do:

1. From the sample city table, write a SELECT query to request the `countrycode` column.
2. Write a second SELECT query to request the `name`, `countrycode`, and `district` columns with the clause, and group the results by `countrycode`.
3. After creating your example, be ready to explain your findings to the class.

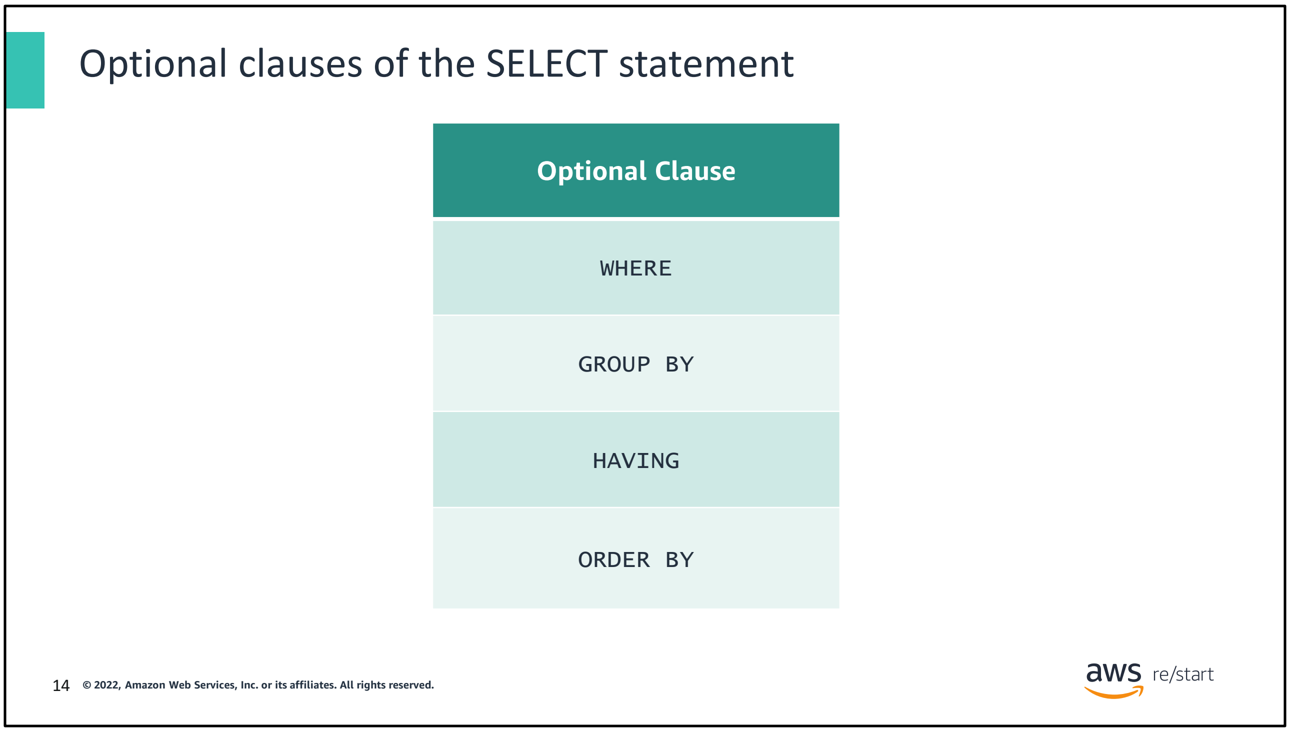
Answers:

```
1. SELECT countrycode
   FROM city;

2. SELECT name, countrycode, district
   FROM city
   GROUP BY countrycode;
```

Optional clauses

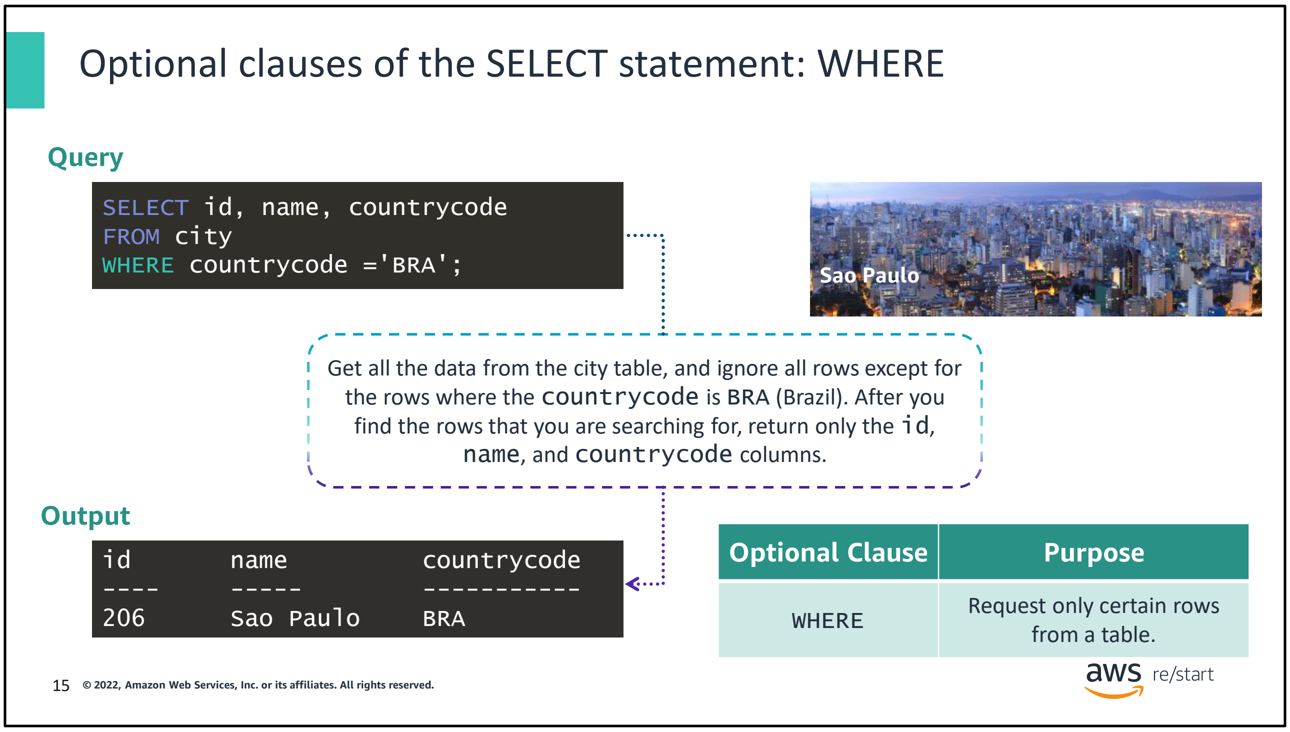
You can use a number of optional clauses with the SELECT statement.



Optional clauses of the SELECT statement

Optional Clause
`WHERE`
`GROUP BY`
`HAVING`
`ORDER BY`

The table lists some optional clauses that you can use with the SELECT statement. The next few slides briefly cover the use case and syntax for each of the optional clauses.



Optional clauses of the SELECT statement: WHERE

Query

```
SELECT id, name, countrycode
FROM city
WHERE countrycode ='BRA';
```

Get all the data from the city table, and ignore all rows except for the rows where the `countrycode` is BRA (Brazil). After you find the rows that you are searching for, return only the `id`, `name`, and `countrycode` columns.

Output

```
id      name         countrycode
----    -----        -----------
206     Sao Paulo    BRA
```

Optional Clause	Purpose
WHERE	Request only certain rows from a table.

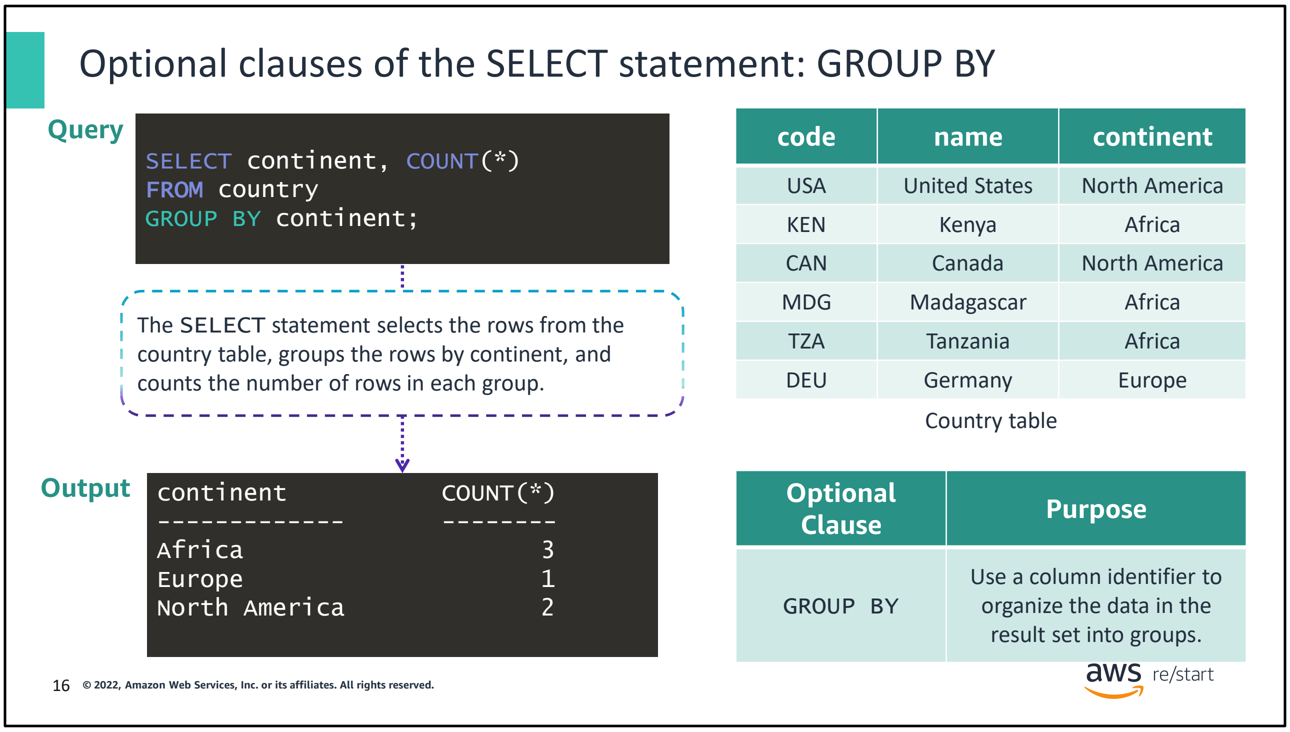
In SQL, you can use the `WHERE` clause to apply a filter that selects only certain rows from a table. In a `SELECT` statement, the WHERE clause is optional. The `SELECT-FROM-WHERE` block can be useful for locating certain information in rows. You could use this construct if you needed a list of all the cities that are located within a country.

For this example, the following is the request: Get all the data from the city table, and ignore all rows except the rows where the `countrycode` is BRA (Brazil). After you find the rows that you are searching for, return only the `id`, `name`, and `countrycode` columns.

The SQL query is as follows:

```
SELECT id, name, countrycode
FROM city
WHERE countrycode='BRA';
```

Note: The output that is shown is partial and does not list all of the rows that contain BRA in the table.



Optional clauses of the SELECT statement: GROUP BY

Query

```
SELECT continent, COUNT(*)
FROM country
GROUP BY continent;
```

The SELECT statement selects the rows from the country table, groups the rows by continent, and counts the number of rows in each group.

Output

```
continent           COUNT(*)
-------------       --------
Africa                     3
Europe                     1
North America              2
```

code	name	continent
USA	United States	North America
KEN	Kenya	Africa
CAN	Canada	North America
MDG	Madagascar	Africa
TZA	Tanzania	Africa
DEU	Germany	Europe

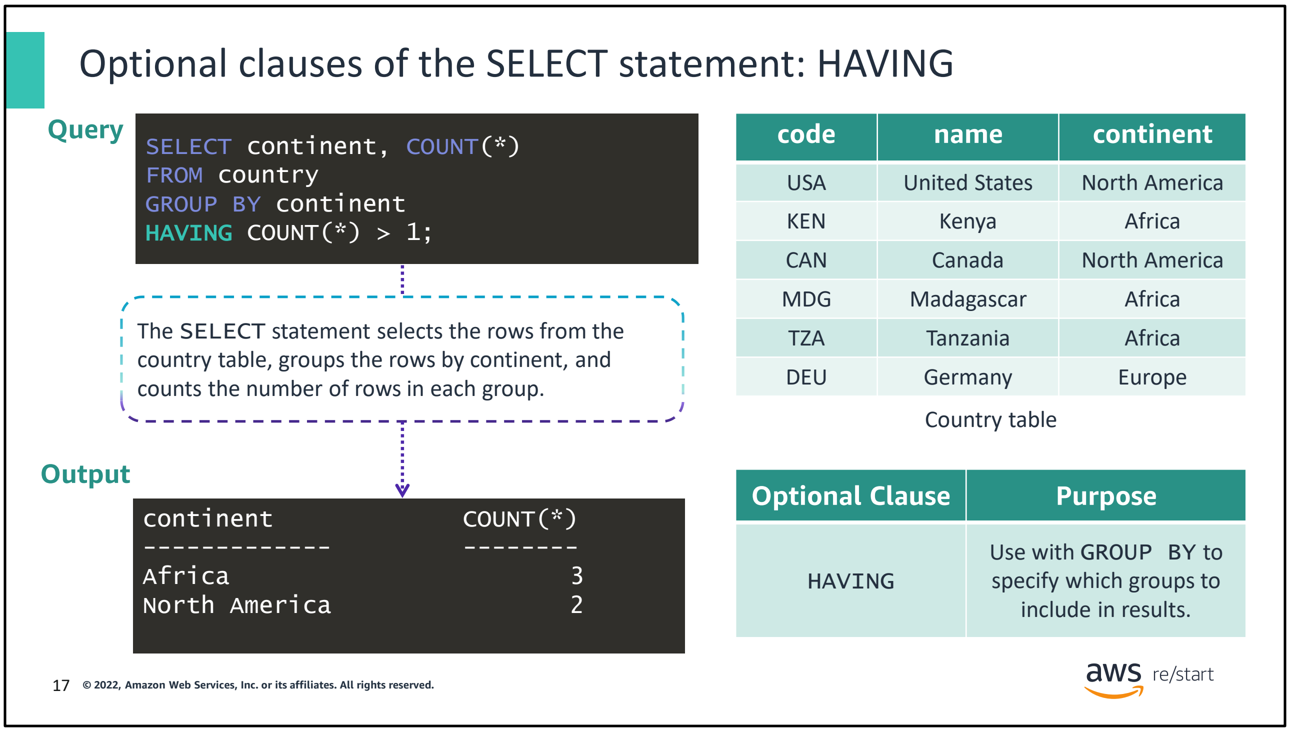
Country table

Optional Clause	Purpose
GROUP BY	Use a column identifier to organize the data in the result set into groups.

aws re/start

Here, the SELECT statement selects the rows from the country table, groups the rows by `continent`, and counts the number of rows in each group. The result is a listing of the number of countries in each continent.

Notice that the GROUP BY clause typically requires an aggregate function in the SELECT clause. In this case, the COUNT() aggregate function is used to count the number of rows in a table.



Optional clauses of the SELECT statement: HAVING

Query

```
SELECT continent, COUNT(*)
FROM country
GROUP BY continent
HAVING COUNT(*) > 1;
```

The SELECT statement selects the rows from the country table, groups the rows by continent, and counts the number of rows in each group.

Output

```
continent                COUNT(*)
-------------            --------
Africa                          3
North America                   2
```

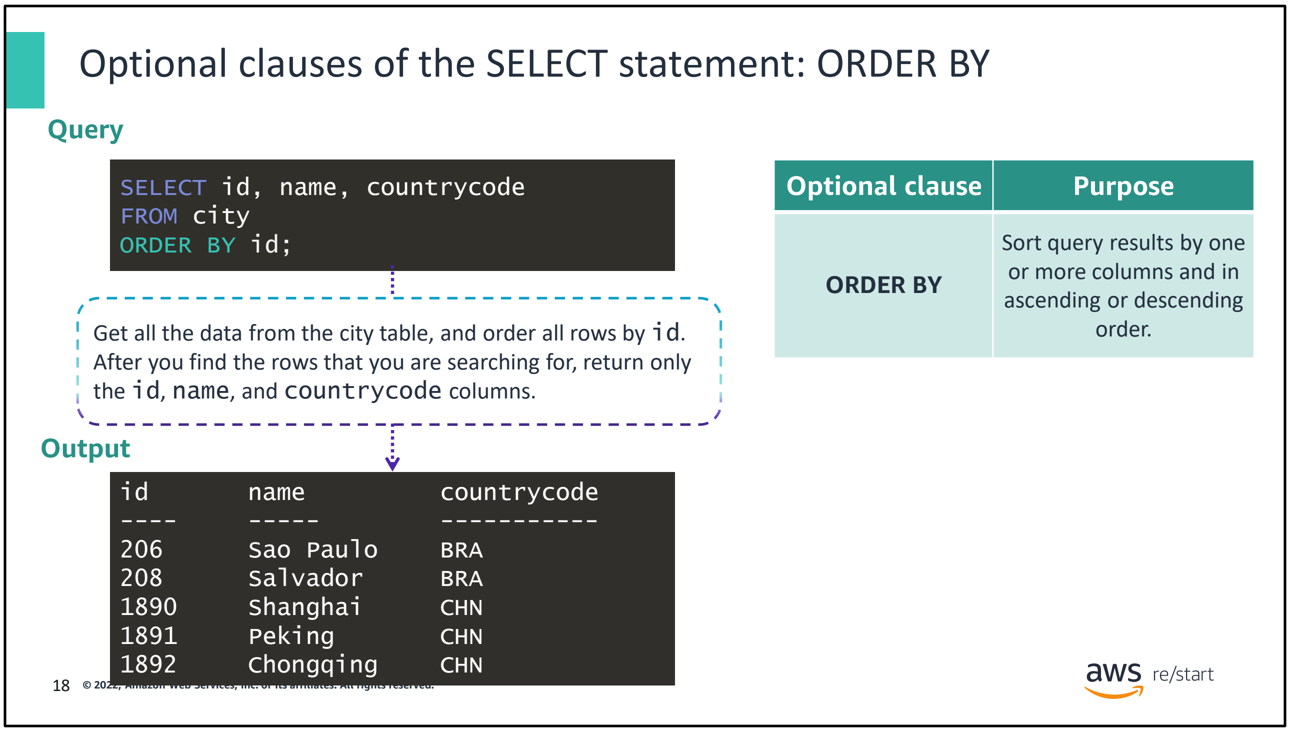
code	name	continent
USA	United States	North America
KEN	Kenya	Africa
CAN	Canada	North America
MDG	Madagascar	Africa
TZA	Tanzania	Africa
DEU	Germany	Europe

Country table

Optional Clause	Purpose
HAVING	Use with GROUP BY to specify which groups to include in results.

aws re/start

The HAVING clause filters the results of a GROUP BY clause in a SELECT statement. In this example, the query selects only the continents that have more than one country after the rows in the table are grouped by continent.



Optional clauses of the SELECT statement: ORDER BY

Query

```
SELECT id, name, countrycode
FROM city
ORDER BY id;
```

Get all the data from the city table, and order all rows by `id`. After you find the rows that you are searching for, return only the `id`, `name`, and `countrycode` columns.

Output

```
id        name           countrycode
----      -----          -----------
206       Sao Paulo      BRA
208       Salvador       BRA
1890      Shanghai       CHN
1891      Peking         CHN
1892      Chongqing      CHN
```

Optional clause	Purpose
ORDER BY	Sort query results by one or more columns and in ascending or descending order.

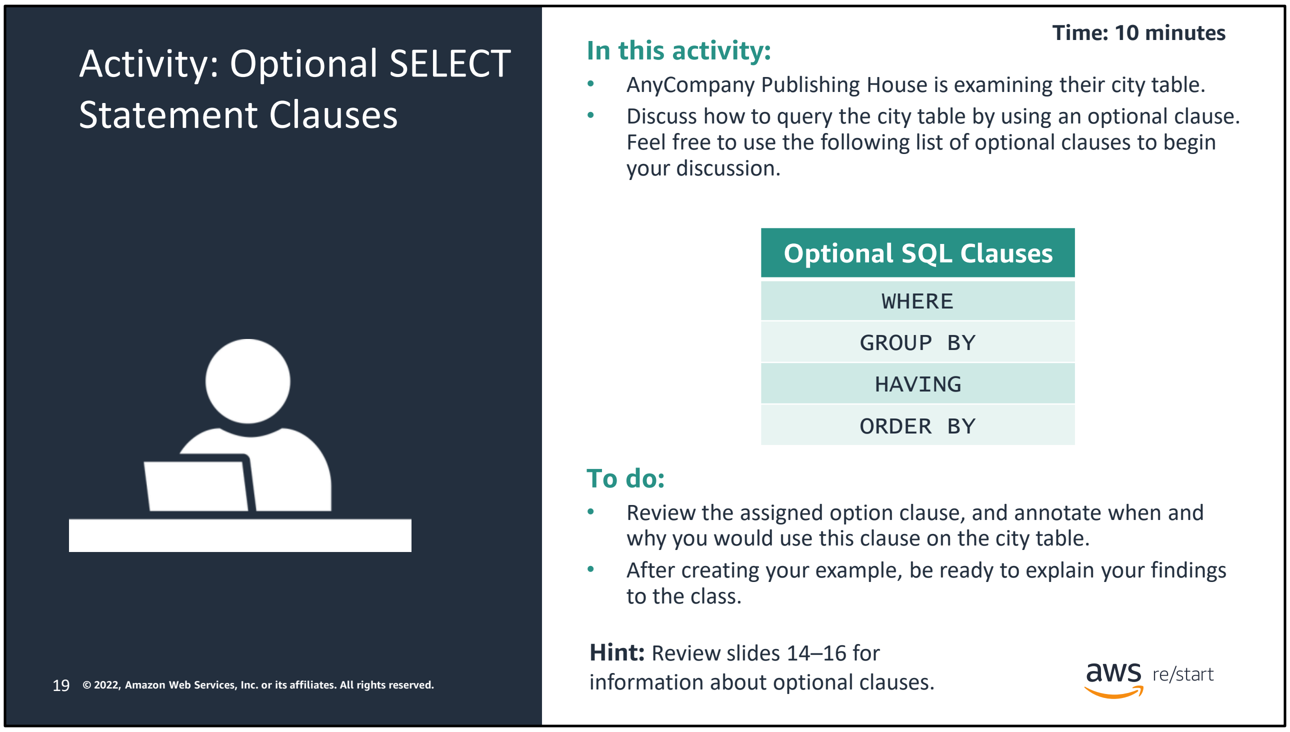
Use the ORDER BY clause to sort query results by one or more columns and in ascending or descending order. If the items in the table are needed in a specific order of importance, you might need to order the results in ascending or descending order.

This example makes the following request: Get all the data from the city table, and order all rows by `id`. After you find the rows that you are searching for, return only the `id`, `name`, and `countrycode` columns.

The SQL query is as follows:

```
SELECT id, name, countrycode
FROM city
ORDER BY id;
```

Note: The output that is shown is partial and does not list all of the rows in the table that the query would normally return.



Activity: Optional SELECT Statement Clauses

Time: 10 minutes

In this activity:

- AnyCompany Publishing House is examining their city table.
- Discuss how to query the city table by using an optional clause. Feel free to use the following list of optional clauses to begin your discussion.

Optional SQL Clauses
`WHERE`
`GROUP BY`
`HAVING`
`ORDER BY`

To do:

- Review the assigned option clause, and annotate when and why you would use this clause on the city table.
- After creating your example, be ready to explain your findings to the class.

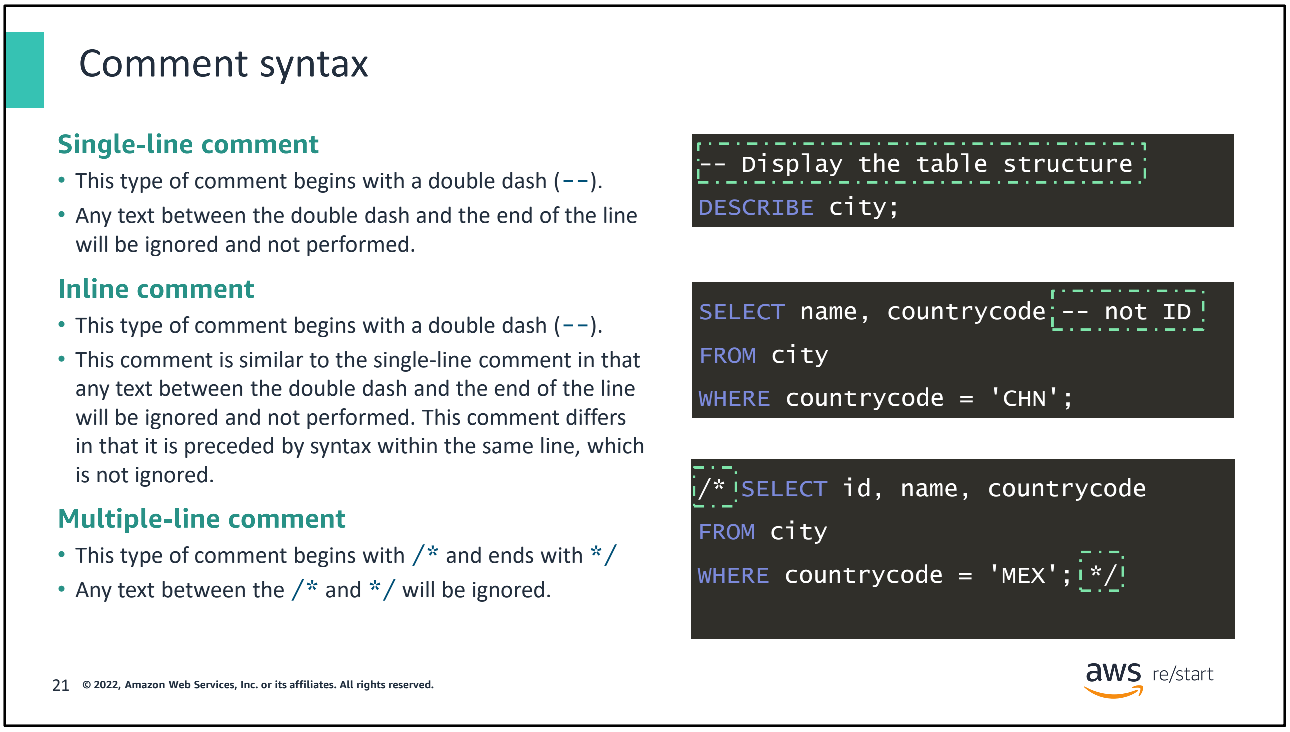
Hint: Review slides 14–16 for information about optional clauses.

AnyCompany Publishing House is examining their city table. They want to query the table.

1. The class will be split into four groups. Each group will be assigned an optional clause.
2. Within your group, discuss when and why you would use your assigned clause.
3. After the activity is complete, discuss the results of your findings with the rest of the class.

Comments

You can also add comments within SQL to clarify statements or clauses.



Comment syntax

Single-line comment

- This type of comment begins with a double dash (--).
- Any text between the double dash and the end of the line will be ignored and not performed.

Inline comment

- This type of comment begins with a double dash (--).
- This comment is similar to the single-line comment in that any text between the double dash and the end of the line will be ignored and not performed. This comment differs in that it is preceded by syntax within the same line, which is not ignored.

Multiple-line comment

- This type of comment begins with /* and ends with */
- Any text between the /* and */ will be ignored.

```
-- Display the table structure
DESCRIBE city;
```

```
SELECT name, countrycode -- not ID
FROM city
WHERE countrycode = 'CHN';
```

```
/* SELECT id, name, countrycode
FROM city
WHERE countrycode = 'MEX'; */
```

Comments begin with specific characters to denote that they are to be ignored and not run. This example shows different ways to write comments in SQL.

Single-line comment

- This type of comment begins with a double dash (--).
- Any text between the double dash and the end of the line will be ignored and not performed.

```
-- Display the table structure
DESCRIBE city;
```

Inline comment

- This type of comment begins with a double dash (--).
- This comment is similar to the single-line comment in that any text between the double dash and the end of the line will be ignored and not performed. This comment differs in that it is preceded by syntax within the same line that is not ignored.

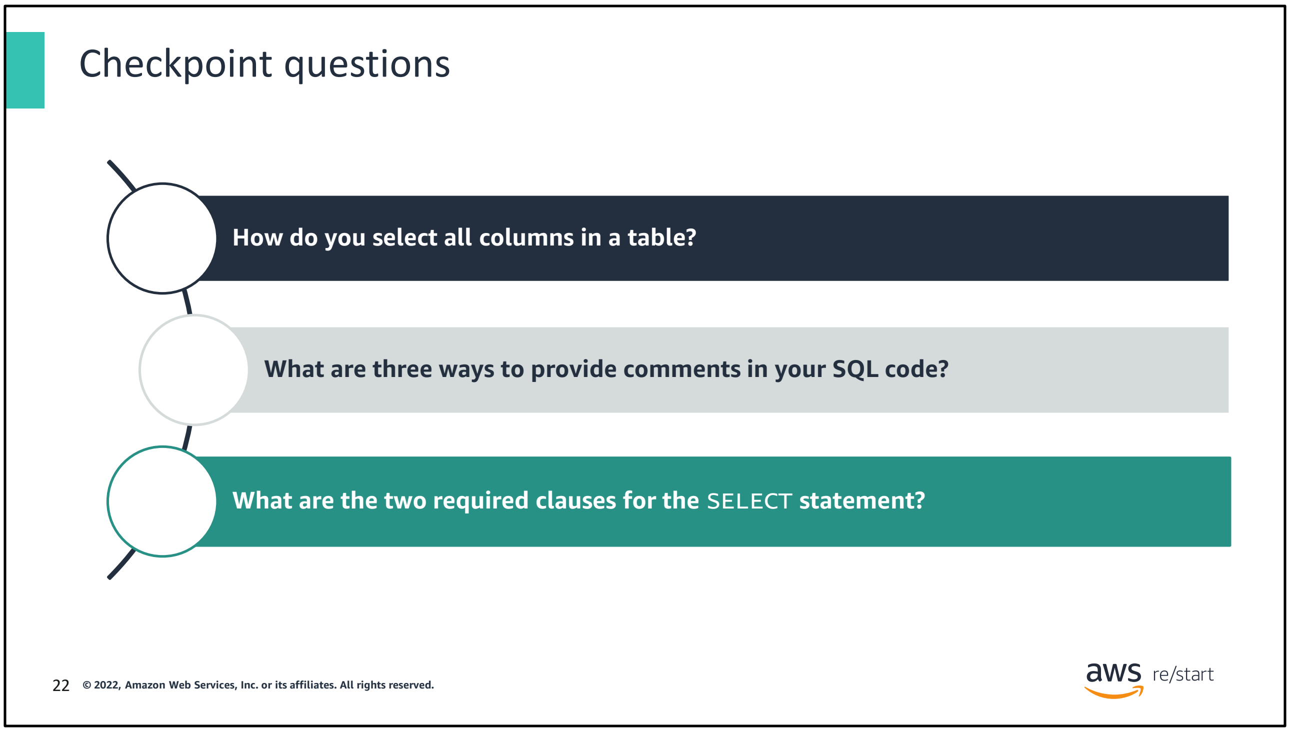
```
SELECT name, countrycode -- not ID
```

```
FROM city
WHERE countrycode = 'CHN';
```

Multiple-line comment

- This type of comment begins with /* and ends with */
- Any text between the /* and */ will be ignored.

```
/* SELECT id, name, countrycode
FROM city
WHERE countrycode = 'MEX'; */
```



Checkpoint questions

How do you select all columns in a table?

What are three ways to provide comments in your SQL code?

What are the two required clauses for the SELECT statement?

1. How do you select all columns in a table?

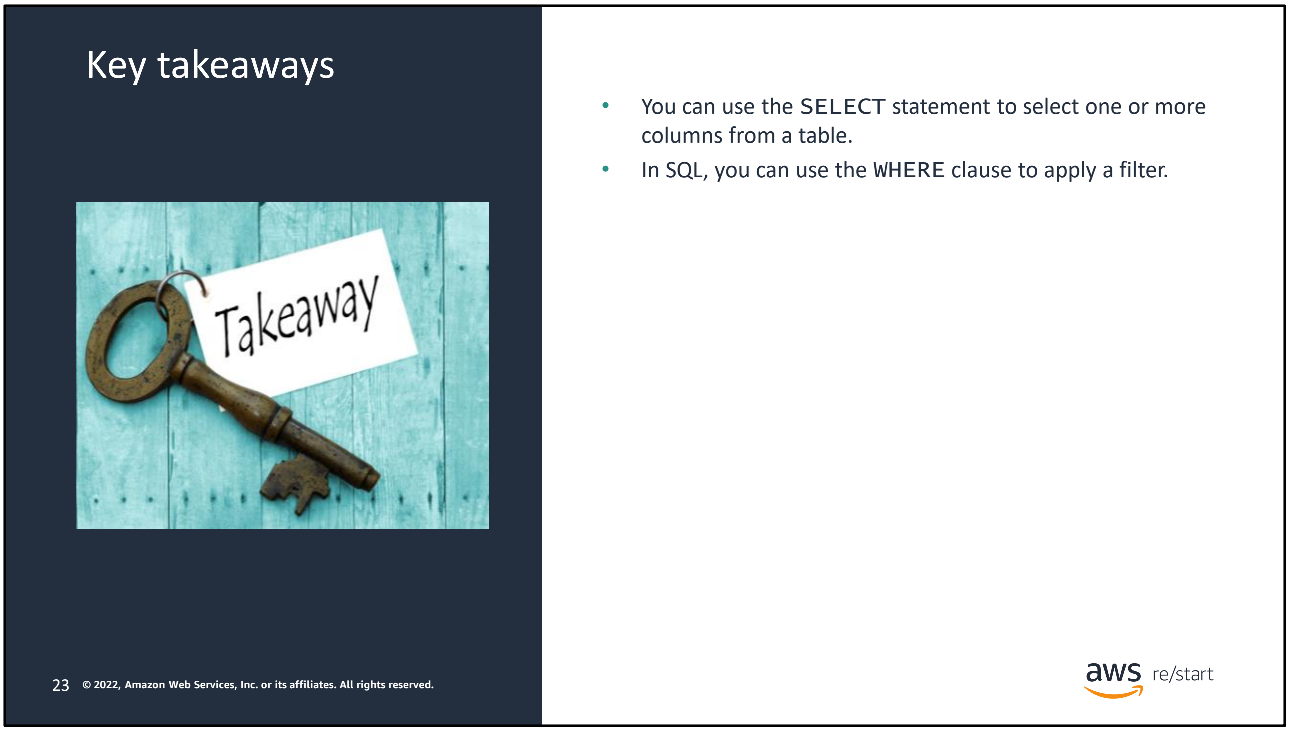
   Use the SELECT * statement.

2. What are three ways to provide comments in your SQL code?

   - Single-line comments
   - Multi-line comments
   - Inline comments

3. What are the two required clauses for the SELECT statement?

   The SELECT clause with column names and the FROM clause with the table name

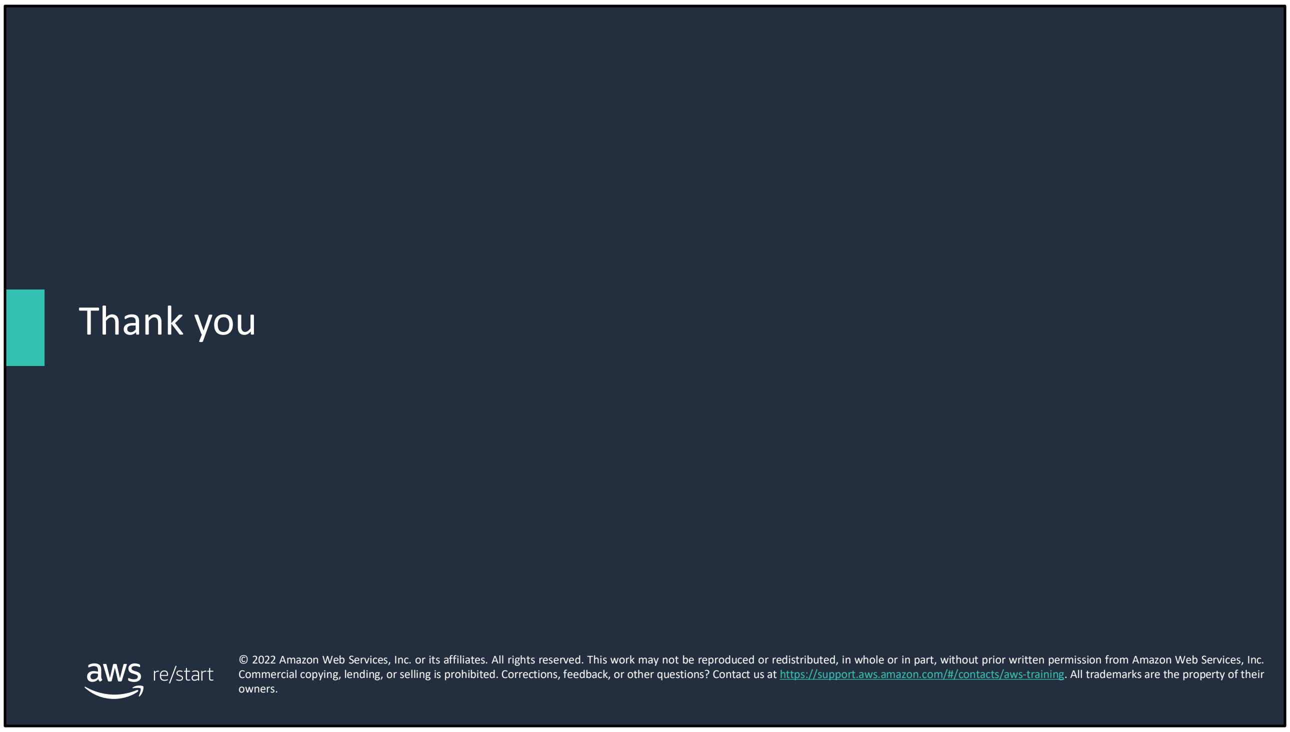


Key takeaways

- You can use the `SELECT` statement to select one or more columns from a table.
- In SQL, you can use the `WHERE` clause to apply a filter.

This module includes the following key takeaways:

- You can use the `SELECT` statement to select one or more columns from a table.
- In SQL, you can use the `WHERE` clause to apply a filter.



Thank you

Thank you for completing this module.