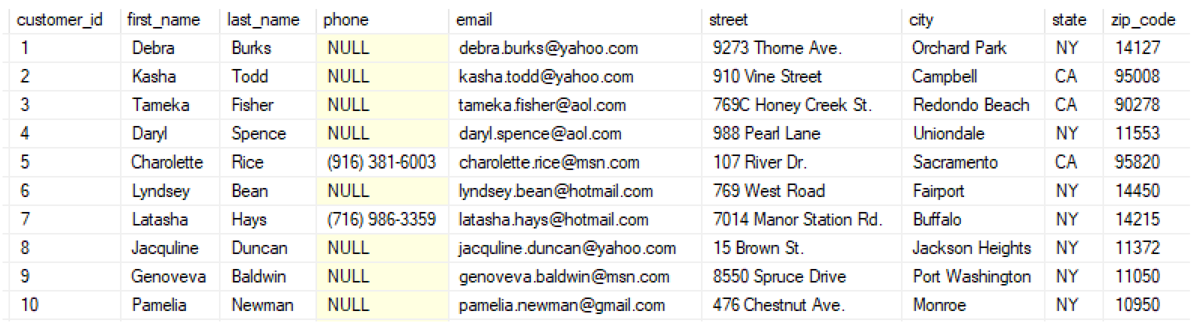
LAB - Select Statement

This lab helps you learn how to query data from the SQL Server database. We will start with a simple query that allows you to retrieve data from a single table.

Database tables are objects that stores all the data in a database. In a table, data is logically organized in a row-and-column format which is similar to a spreadsheet.

In a table, each row represents a unique record and each column represents a field in the record. For example, the `customers` table contains customer data such as customer identification number, first name, last name, phone, email, and address information as shown below:



customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553
5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	NY	14450
7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	NY	14215
8	Jacquline	Duncan	NULL	jacquline.duncan@yahoo.com	15 Brown St.	Jackson Heights	NY	11372
9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	NY	11050
10	Pamelia	Newman	NULL	pamelia.newman@gmail.com	476 Chestnut Ave.	Monroe	NY	10950

SQL Server uses schemas to logically groups tables and other database objects. In our sample database, we have two schemas: `sales` and `production`. The `sales` schema groups all the sales related tables while the `production` schema groups all the production related tables.

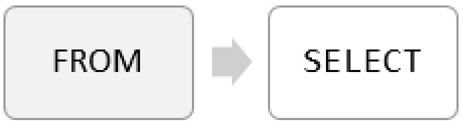
To query data from a table, you use the `SELECT` statement. The following illustrates the most basic form of the `SELECT` statement:

```
SELECT
    select_list
FROM
    schema_name.table_name;
```

In this syntax:

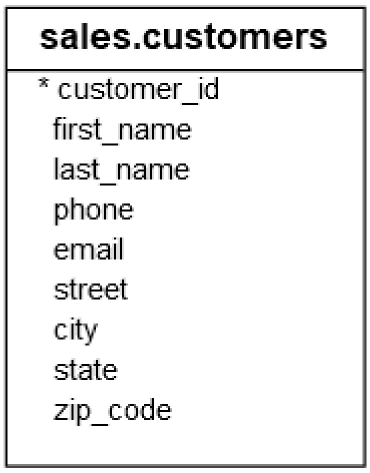
- First, specify a list of comma-separated columns from which you want to query data in the `SELECT` clause.
- Second, specify the source table and its schema name on the `FROM` clause.

When processing the `SELECT` statement, SQL Server processes the `FROM` clause first and then the `SELECT` clause even though the `SELECT` clause appears first in the query.



SQL Server `SELECT` statement examples

Let's use the `customers` table in the sample database for the demonstration.



sales.customers

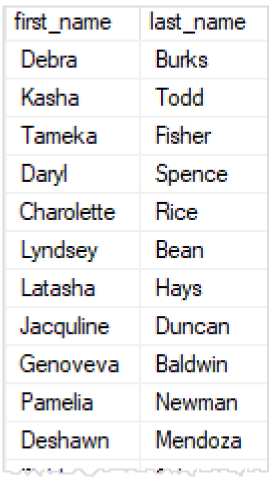
* customer_id
first_name
last_name
phone
email
street
city
state
zip_code

A) `SELECT` – retrieve some columns of a table:

The following query finds the first name and last name of all customers:

```
SELECT
    first_name,
    last_name
FROM
    sales.customers;
```

Here is the result:

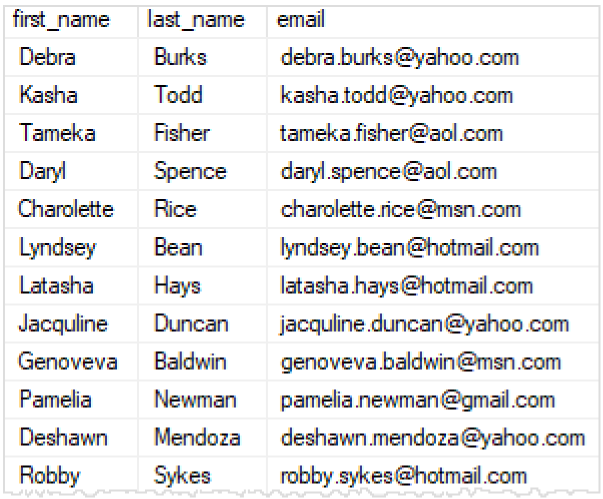


first_name	last_name
Debra	Burks
Kasha	Todd
Tameka	Fisher
Daryl	Spence
Charolette	Rice
Lyndsey	Bean
Latasha	Hays
Jacquline	Duncan
Genoveva	Baldwin
Pamelia	Newman
Deshawn	Mendoza
[illegible]	[illegible]

The result of a query is called a result set.

The following statement returns the first names, last names, and emails of all customers:

```
SELECT
    first_name,
    last_name,
    email
FROM
    sales.customers;
```

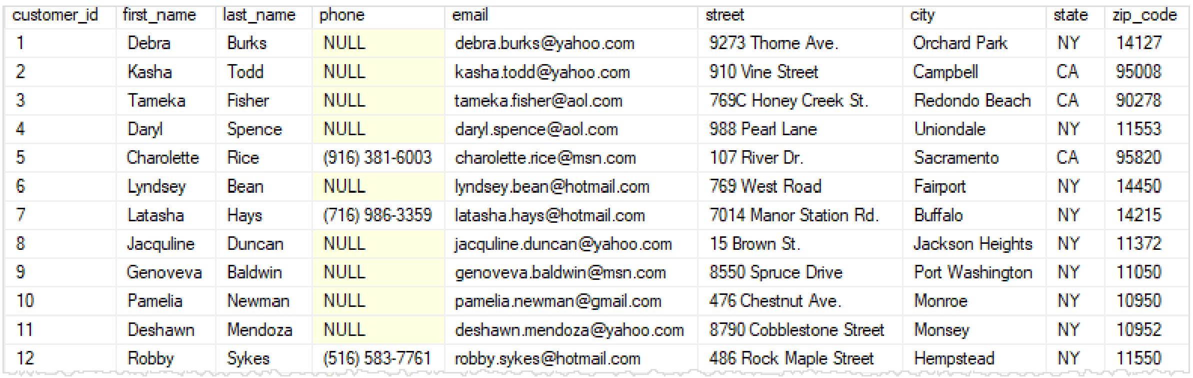


first_name	last_name	email
Debra	Burks	debra.burks@yahoo.com
Kasha	Todd	kasha.todd@yahoo.com
Tameka	Fisher	tameka.fisher@aol.com
Daryl	Spence	daryl.spence@aol.com
Charolette	Rice	charolette.rice@msn.com
Lyndsey	Bean	lyndsey.bean@hotmail.com
Latasha	Hays	latasha.hays@hotmail.com
Jacquline	Duncan	jacquline.duncan@yahoo.com
Genoveva	Baldwin	genoveva.baldwin@msn.com
Pamelia	Newman	pamelia.newman@gmail.com
Deshawn	Mendoza	deshawn.mendoza@yahoo.com
Robby	Sykes	robby.sykes@hotmail.com

B) `SELECT` – retrieve all columns from a table

To get data from all columns of a table, you can specify all the columns in the select list. You can also use `SELECT *` as a shorthand to save some typing:

```
SELECT
    *
FROM
    sales.customers;
```



customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553
5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	NY	14450
7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	NY	14215
8	Jacquline	Duncan	NULL	jacquline.duncan@yahoo.com	15 Brown St.	Jackson Heights	NY	11372
9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	NY	11050
10	Pamelia	Newman	NULL	pamelia.newman@gmail.com	476 Chestnut Ave.	Monroe	NY	10950
11	Deshawn	Mendoza	NULL	deshawn.mendoza@yahoo.com	8790 Cobblestone Street	Monsey	NY	10952
12	Robby	Sykes	(516) 583-7761	robby.sykes@hotmail.com	486 Rock Maple Street	Hempstead	NY	11550

The `SELECT *` is useful for examining the columns and data of a table that you are not familiar with. It is also helpful for ad-hoc queries.

However, you should not use the `SELECT *` for real production code due to the following main reasons:

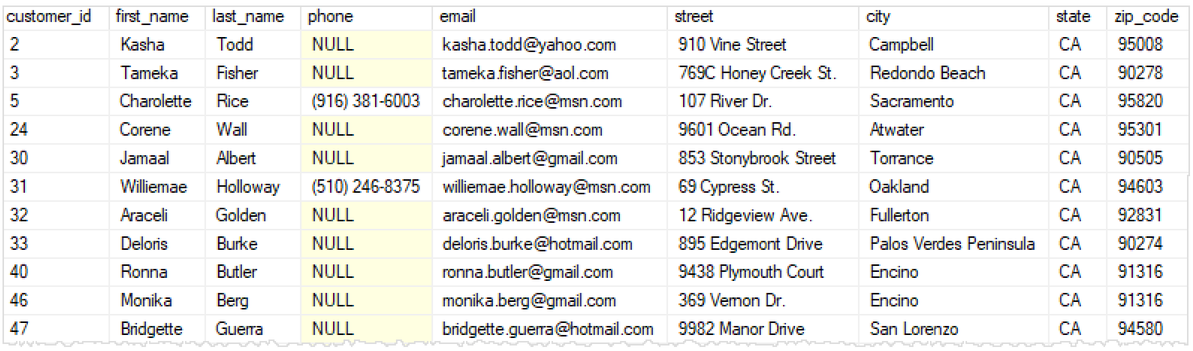
1. First, `SELECT *` often retrieves more data than your application needs to function. It causes unnecessary data to transfer from the SQL Server to the client application, taking more time for data to travel across the network and slowing down the application.

2. Second, if the table is added one or more new columns, the `SELECT *` just retrieves all columns that include the newly added columns which were not intended for use in the application. This could make the application crash.

C) `SELECT` – sort the result set

To filter rows based on one or more conditions, you use a `WHERE` clause as shown in the following example:

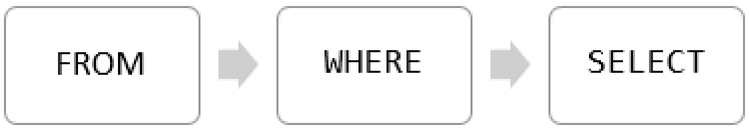
```
SELECT
    *
FROM
    sales.customers
WHERE
    state = 'CA';
```



customer_id	first_name	last_name	phone	email	street	city	state	zip_code
2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
24	Corene	Wall	NULL	corene.wall@msn.com	9601 Ocean Rd.	Atwater	CA	95301
30	Jamaal	Albert	NULL	jamaal.albert@gmail.com	853 Stonybrook Street	Torrance	CA	90505
31	Williemae	Holloway	(510) 246-8375	williemae.holloway@msn.com	69 Cypress St.	Oakland	CA	94603
32	Araceli	Golden	NULL	araceli.golden@msn.com	12 Ridgeview Ave.	Fullerton	CA	92831
33	Deloris	Burke	NULL	deloris.burke@hotmail.com	895 Edgemont Drive	Palos Verdes Peninsula	CA	90274
40	Ronna	Butler	NULL	ronna.butler@gmail.com	9438 Plymouth Court	Encino	CA	91316
46	Monika	Berg	NULL	monika.berg@gmail.com	369 Vernon Dr.	Encino	CA	91316
47	Bridgette	Guerra	NULL	bridgette.guerra@hotmail.com	9982 Manor Drive	San Lorenzo	CA	94580

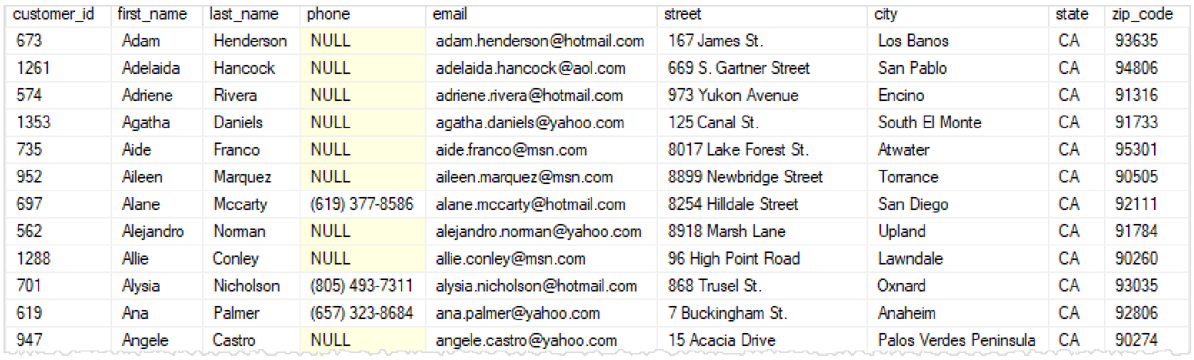
In this example, the query returns the customers who locate in California.

When the `WHERE` clause is available, SQL Server processes the clauses of the query in the following sequence: `FROM`, `WHERE`, and `SELECT`.



To sort the result set based on one or more columns, you use the `ORDER BY` clause as shown in the following example:

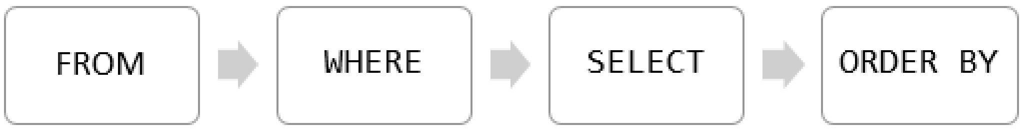
```
SELECT
    *
FROM
    sales.customers
WHERE
    state = 'CA'
ORDER BY
    first_name;
```



customer_id	first_name	last_name	phone	email	street	city	state	zip_code
673	Adam	Henderson	NULL	adam.henderson@hotmail.com	167 James St.	Los Banos	CA	93635
1261	Adelaida	Hancock	NULL	adelaida.hancock@aol.com	669 S. Gartner Street	San Pablo	CA	94806
574	Adriene	Rivera	NULL	adriene.rivera@hotmail.com	973 Yukon Avenue	Encino	CA	91316
1353	Agatha	Daniels	NULL	agatha.daniels@yahoo.com	125 Canal St.	South El Monte	CA	91733
735	Aide	Franco	NULL	aide.franco@msn.com	8017 Lake Forest St.	Atwater	CA	95301
952	Aileen	Marquez	NULL	aileen.marquez@msn.com	8899 Newbridge Street	Torrance	CA	90505
697	Alane	Mccarty	(619) 377-8586	alane.mccarty@hotmail.com	8254 Hilldale Street	San Diego	CA	92111
562	Alejandro	Norman	NULL	alejandro.norman@yahoo.com	8918 Marsh Lane	Upland	CA	91784
1288	Allie	Conley	NULL	allie.conley@msn.com	96 High Point Road	Lawndale	CA	90260
701	Alysia	Nicholson	(805) 493-7311	alysia.nicholson@hotmail.com	868 Trusel St.	Oxnard	CA	93035
619	Ana	Palmer	(657) 323-8684	ana.palmer@yahoo.com	7 Buckingham St.	Anaheim	CA	92806
947	Angele	Castro	NULL	angele.castro@yahoo.com	15 Acacia Drive	Palos Verdes Peninsula	CA	90274

In this example, the `ORDER BY` clause sorts the customers by their first names in ascending order.

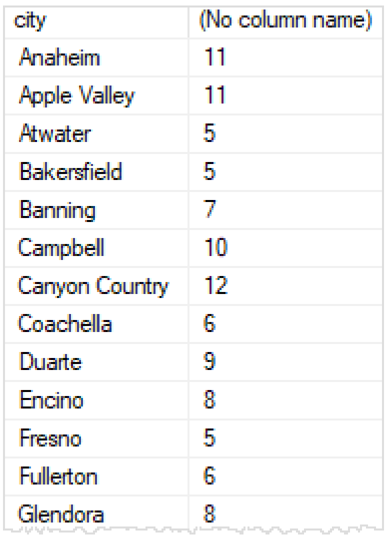
In this case, SQL Server processes the clauses of the query in the following sequence: `FROM`, `WHERE`, `SELECT`, and `ORDER BY`.



D) `SELECT` – combine rows into groups example

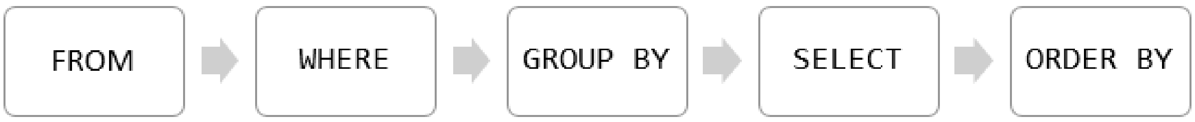
To group rows into groups, you use the `GROUP BY` clause. For example, the following statement returns all the cites of customers located in California and the number of customers in each city.

```
SELECT
    city,
    COUNT (*)
FROM
    sales.customers
WHERE
    state = 'CA'
GROUP BY
    city
ORDER BY
    city;
```



city	(No column name)
Anaheim	11
Apple Valley	11
Atwater	5
Bakersfield	5
Banning	7
Campbell	10
Canyon Country	12
Coachella	6
Duarte	9
Encino	8
Fresno	5
Fullerton	6
Glendora	8

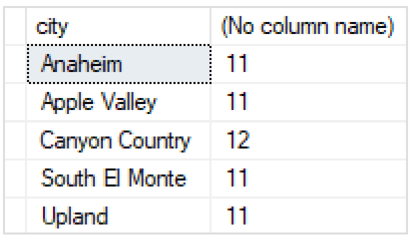
In this case, SQL Server processes the clauses in the following sequence: `FROM`, `WHERE`, `GROUP BY`, `SELECT`, and `ORDER BY`.



E) `SELECT` – filter groups example

To filter groups based on one or more conditions, you use the `HAVING` clause. The following example returns the city in California which has more than 10 customers:

```
SELECT
    city,
    COUNT (*)
FROM
    sales.customers
WHERE
    state = 'CA'
GROUP BY
    city
HAVING
    COUNT (*) > 10
ORDER BY
    city;
```



city	(No column name)
Anaheim	11
Apple Valley	11
Canyon Country	12
South El Monte	11
Upland	11

Notice that the `WHERE` clause filters rows while the `HAVING` clause filter groups.

In this lab, you have learned how to use the SQL Server `SELECT` statement to query data from a single table.