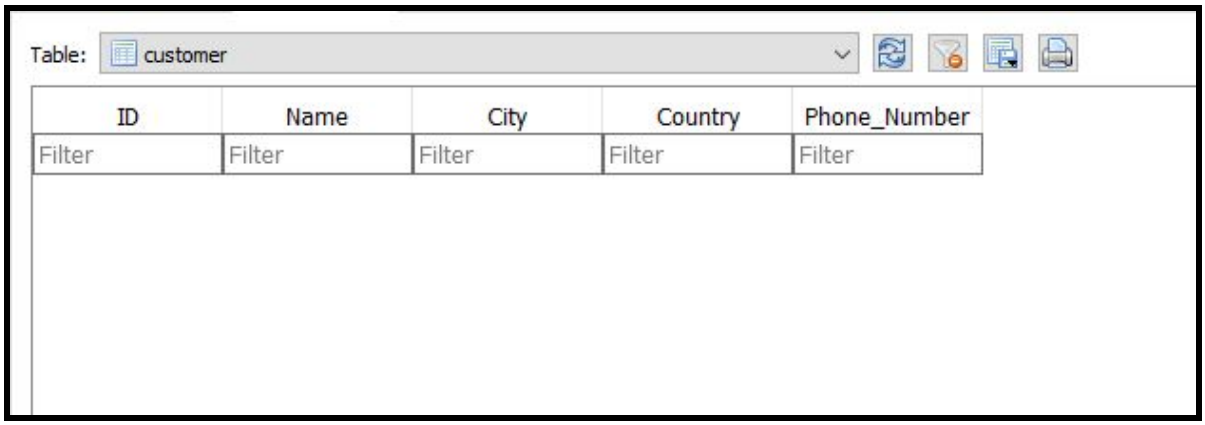
Create table:

- Let's say we want to create a customer table such that the relational schema is this customer(ID, name, city, country, phone_number)
- The code is this: **CREATE TABLE customer('ID' INTEGER PRIMARY KEY AUTOINCREMENT, 'Name' TEXT, 'City' TEXT, 'Country' TEXT, 'Phone_Number' TEXT);**

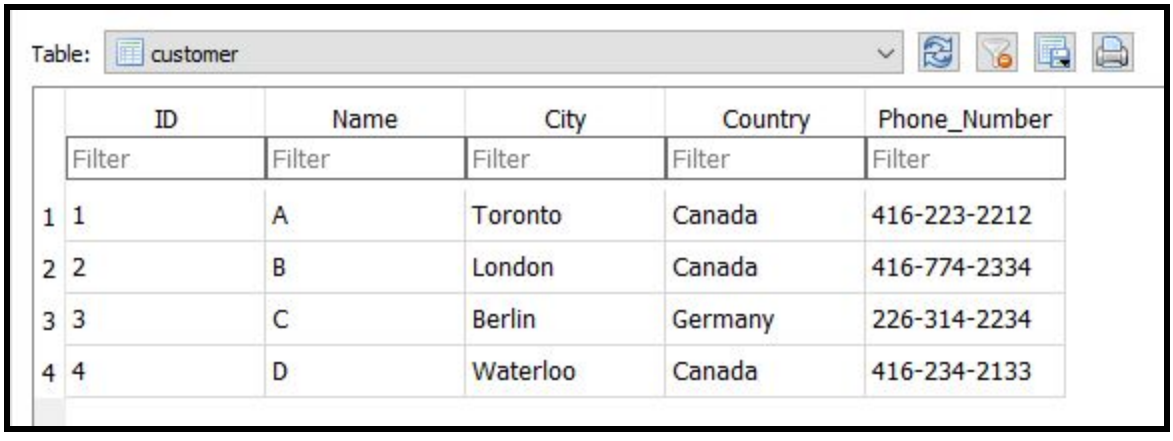
Insert Into:

- To add information into a table, we use the insert into command.
- Right now, the customer table is empty, as shown below.



- If we run the below commands, we get the following table:

INSERT INTO customer(Name, City, Country, Phone_Number) VALUES("A", "Toronto", "Canada", "416-223-2212");
INSERT INTO customer(name, city, country, phone_number) VALUES ('B', 'London', 'Canada', '416-774-2334');
INSERT INTO customer (name, city, country, phone_number) VALUES ('C', 'Berlin', 'Germany', '226-314-2234');
INSERT INTO customer(name, city, country, phone_number) VALUES ('D', 'Waterloo', 'Canada', '416-234-2133');



	ID	Name	City	Country	Phone_Number
	Filter	Filter	Filter	Filter	Filter
1	1	A	Toronto	Canada	416-223-2212
2	2	B	London	Canada	416-774-2334
3	3	C	Berlin	Germany	226-314-2234
4	4	D	Waterloo	Canada	416-234-2133

Select:

- To get all the columns of the customer relation, we'd run this command:
  SELECT * FROM customer;
  The output table looks like this:

	ID	Name	City	Country	Phone_Number
1	1	A	Toronto	Canada	416-223-2212
2	2	B	London	Canada	416-774-2334
3	3	C	Berlin	Germany	226-314-2234
4	4	D	Waterloo	Canada	416-234-2133

- To get the column "Country", we'd run this command:
  SELECT country FROM customer;
  The output table looks like this:

	Country
1	Canada
2	Canada
3	Germany
4	Canada

- To get all the customers from Canada, we'd run this command:
  SELECT * FROM customer WHERE country IS 'Canada';
  The output table looks like this:

	ID	Name	City	Country	Phone_Number
1	1	A	Toronto	Canada	416-223-2212
2	2	B	London	Canada	416-774-2334
3	4	D	Waterloo	Canada	416-234-2133

- To get the ID column from the table, but rename is customerID, we'd run this command:
  SELECT ID AS CUSTOMERID FROM customer;
  The output table looks like this:

	CUSTOMERID
1	1
2	2
3	3
4	4

Select Distinct:

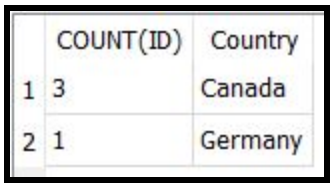
- To get all the different countries in the column "Country", we'd run this command:
  SELECT DISTINCT country FROM customer;

The output table looks like this:

	Country
1	Canada
2	Germany

Group by, Count, Having, Max, Min, Order By, Limit:

- If we want to get the number of customers in each country, we'd run this command: **SELECT COUNT(ID), Country FROM customer GROUP BY Country;**
  The output table looks like this:



	COUNT(ID)	Country
1	3	Canada
2	1	Germany

- If we want to get how many customers are in Canada, we'd run this command: **SELECT COUNT(ID), Country FROM customer GROUP BY Country HAVING Country IS 'Canada';**
  The output table looks like this:

	COUNT(ID)	Country
1	3	Canada

- If we want to get the country that has the maximum number of customers, we'd run this command: **SELECT country FROM (SELECT COUNT(ID) mycount, Country FROM customer GROUP BY Country) ORDER BY mycount DESC LIMIT 1;**
  The output table looks like this:

	Country
1	Canada

- If we want to get the country that has the least number of customers, we'd run this command:
  SELECT country FROM (SELECT COUNT(ID) mycount, country FROM customer GROUP BY country) ORDER BY mycount LIMIT 1;
  The output table looks like this:

	country
1	Germany

Update:

- If we want to change the phone number of customer A to "416-223-2222", we'd run this command: **UPDATE customer SET Phone_Number = '416-223-2222' where ID = 1;**

The customer relation now looks like this:

	ID	Name	City	Country	Phone_Number
	Filter	Filter	Filter	Filter	Filter
1	1	A	Toronto	Canada	416-223-2222
2	2	B	London	Canada	416-774-2334
3	3	C	Berlin	Germany	226-314-2234
4	4	D	Waterloo	Canada	416-234-2133

Suppose we have the following 2 relations:

Table: Student

	ID	name	address
	Filter	Filter	Filter
1	1	A	48 XYZ Drive
2	2	B	10 ABC Road
3	3	C	20 ABC Ave
4	4	D	20 ABC Ave
5	5	E	3452 XYZ Drive
6	7	G	1234 X Ave

and

Table: marks

	ID	mark	student_name	class
	Filter	Filter	Filter	Filter
1	1	85	A	science
2	2	67	B	math
3	3	90	C	math
4	4	56	D	math
5	5	85	E	english
6	6	75	F	french

Cross Join:

- To get a cartesian product of the student and marks relations, I'd run the command:
  SELECT * FROM Student CROSS JOIN Marks;
  The output relation is this:

ID	name	address	ID	mark	student_name	class
1	A	48 XYZ Drive	1	85	A	science
1	A	48 XYZ Drive	2	67	B	math
1	A	48 XYZ Drive	3	90	C	math
1	A	48 XYZ Drive	4	56	D	math
1	A	48 XYZ Drive	5	85	E	english
1	A	48 XYZ Drive	6	75	F	french
2	B	10 ABC Road	1	85	A	science
2	B	10 ABC Road	2	67	B	math
2	B	10 ABC Road	3	90	C	math
2	B	10 ABC Road	4	56	D	math
2	B	10 ABC Road	5	85	E	english

2	B	10 ABC Road	6	75	F	french
3	C	20 ABC Ave	1	85	A	science
3	C	20 ABC Ave	2	67	B	math
3	C	20 ABC Ave	3	90	C	math
3	C	20 ABC Ave	4	56	D	math
3	C	20 ABC Ave	5	85	E	english
3	C	20 ABC Ave	6	75	F	french
4	D	20 ABC Ave	1	85	A	science
4	D	20 ABC Ave	2	67	B	math
4	D	20 ABC Ave	3	90	C	math
4	D	20 ABC Ave	4	56	D	math
4	D	20 ABC Ave	5	85	E	english
4	D	20 ABC Ave	6	75	F	french
5	E	3452 XYZ Drive	1	85	A	science
5	E	3452 XYZ Drive	2	67	B	math
5	E	3452 XYZ Drive	3	90	C	math
5	E	3452 XYZ Drive	4	56	D	math
5	E	3452 XYZ Drive	5	85	E	english
5	E	3452 XYZ Drive	6	75	F	french
7	G	1234 X Ave	1	85	A	science
7	G	1234 X Ave	2	67	B	math
7	G	1234 X Ave	3	90	C	math
7	G	1234 X Ave	4	56	D	math
7	G	1234 X Ave	5	85	E	english
7	G	1234 X Ave	6	75	F	french

Left Join:

- Running the command **SELECT * FROM student LEFT JOIN marks ON student.id = marks.id;** gets me the following relation:

	ID	name	address	ID	mark	student_name	class
1	1	A	48 XYZ Drive	1	85	A	science
2	2	B	10 ABC Road	2	67	B	math
3	3	C	20 ABC Ave	3	90	C	math
4	4	D	20 ABC Ave	4	56	D	math
5	5	E	3452 XYZ Drive	5	85	E	english
6	7	G	1234 X Ave	*NULL*	*NULL*	*NULL*	*NULL*

- Running the command **SELECT * FROM marks LEFT JOIN student ON student.id = marks.id;** gets me the following relation:

	ID	mark	student_name	class	ID	name	address
1	1	85	A	science	1	A	48 XYZ Drive
2	2	67	B	math	2	B	10 ABC Road
3	3	90	C	math	3	C	20 ABC Ave
4	4	56	D	math	4	D	20 ABC Ave
5	5	85	E	english	5	E	3452 XYZ Drive
6	6	75	F	french	*NULL*	*NULL*	*NULL*

Inner Join:

- Running the command **SELECT * FROM marks INNER JOIN student ON student.id = marks.id;** gets me the following relation:

	ID	mark	student_name	class	ID	name	address
1	1	85	A	science	1	A	48 XYZ Drive
2	2	67	B	math	2	B	10 ABC Road
3	3	90	C	math	3	C	20 ABC Ave
4	4	56	D	math	4	D	20 ABC Ave
5	5	85	E	english	5	E	3452 XYZ Drive

Suppose we have the following 2 relations:

Table: A

	x	y
	Filter	Filter
1	1	a
2	2	b
3	3	c
4	4	d

and

Table: B

	x	y
	Filter	Filter
1	1	x
2	2	b
3	3	c

Union:

- If I run the command **SELECT * FROM A UNION SELECT * FROM B;** I'd get this table:

	x	y
1	1	a
2	1	x
3	2	b
4	3	c
5	4	d

Intersect:

- If I run the command **SELECT * FROM A INTERSECT SELECT * FROM B;** I'd get this table:

	x	y
1	2	b
2	3	c

Except:

- If I run the command **SELECT * FROM A EXCEPT SELECT * FROM B;** I'd get this table:

	x	y
1	1	a
2	4	d

Create View:

- Creating views can be used to break long SQL queries into smaller pieces.
- E.g. Suppose we want to find the names of all students with an average greater than or equal to 85. The relational schemas are given below and the primary ids are underlined:
  student(id, name)
  marks(id, average)

  To do this using views, we would run the following commands:
  CREATE VIEW v1 AS SELECT ID FROM marks WHERE Average >= 85;
  SELECT S.name FROM Student S NATURAL JOIN v1;

  Suppose we tested the above 2 lines on these 2 relations:

Table: Student

	ID	name
	Filter	Filter
1	1	A
2	2	B
3	3	C
4	4	D
5	5	E

and

Table: marks

	ID	Average
	Filter	Filter
1	1	85
2	2	67
3	3	90
4	4	56
5	5	85

The first line, **CREATE VIEW v1 AS SELECT ID FROM marks WHERE Average >= 85;**, would create a view shown below:

Table: v1

	ID
	Filter
1	1
2	3
3	5

The second line, **SELECT S.name FROM Student S NATURAL JOIN v1;**, would create the output table shown below:

	name
1	A
2	C
3	E