DATA MANIPUATION LANGUAGE

INSERT: (MANUAL ENTRY)

QUERY 1: INSERT TWO ROWS OF DATA IN THE EXISTING TABLE customers.

```
INSERT INTO customers(id,first_name,country,score)
VALUES
	(6, 'Anna', 'USA', NULL),
	(7, 'Sam', NULL, 100)
```

	id	first_name	country	score
1	1	Maria	Germany	350
2	2	John	USA	900
3	3	Georg	UK	750
4	4	Martin	Germany	500
5	5	Peter	USA	0
6	6	Anna	USA	NULL
7	7	Sam	NULL	100

Colums and values should be in the same order.

(OR)

```
INSERT INTO customers
VALUES
	(8, 'Annis', 'India', 15)
```

	id	first_name	country	score
1	1	Maria	Germany	350
2	2	John	USA	900
3	3	Georg	UK	750
4	4	Martin	Germany	500
5	5	Peter	USA	0
6	6	Anna	USA	NULL
7	7	Sam	NULL	100
8	8	Annis	India	15

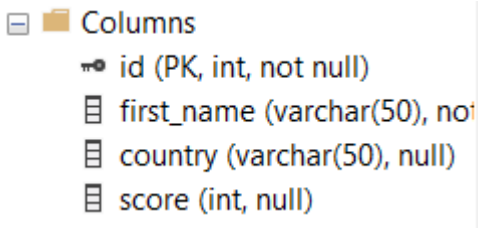
If you want to insert the values in all columns, you don't need specify all the columns, SQL understands the need. But, always list columns for clarity and maintainability.

And you can skip only the columns that are null, if there is no value given by you during INSSERT error will be triggered.

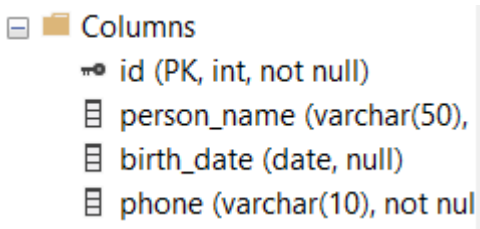
INSERT USING SELECT: (DATA INSERTION FROM ONE TABLE TO ANOTHER TABLE)

QUERY 1: COPY THE DATA FROM TABEL customers TO TABLE persons.

Source Table: customers



Target Table(Empty Table): persons




```
INSERT INTO persons(id, person_name, birth_date, phone)
SELECT
id,
first_name,
NULL,
'Unknown'
FROM customers
```

	id	person_name	birth_date	phone
1	1	Maria	NULL	Unknown
2	2	John	NULL	Unknown
3	3	Georg	NULL	Unknown
4	4	Martin	NULL	Unknown
5	5	Peter	NULL	Unknown
6	6	Anna	NULL	Unknown
7	7	Sam	NULL	Unknown
8	8	Annis	NULL	Unknown

UPDATE: (For changing the data of already existed rows)

Always use WHERE to avoid UPDATING all the rows unintentionally.

QUERY 1: CHANGE THE SCORE OF CUSTOMER 6 TO ZERO.

```
UPDATE customers
SET score =0
WHERE id=6
```

	id	first_name	country	score
1	1	Maria	Germany	350
2	2	John	USA	900
3	3	Georg	UK	750
4	4	Martin	Germany	500
5	5	Peter	USA	0
6	6	Anna	USA	0
7	7	Sam	NULL	100
8	8	Annis	India	15

Without a WHERE al rows will be updated.

BEST PRACTICE : Check with SELECT before running UPDATE to avoid updating wrong data.

```
SELECT * FROM customers
WHERE id=6
```

And also check in the output section how many rows affected. It has to be 1 in our case

QUERY 2: CHANGE THE SCORE OF CUSTOMER 8 TO ZERO AND UPDATE THE COUNTRY TO 'UK'

```
UPDATE customers
SET score=0,
       country = 'UK'
WHERE id=8
SELECT * FROM customers
```

	id	first_name	country	score
1	1	Maria	Germany	350
2	2	John	USA	900
3	3	Georg	UK	750
4	4	Martin	Germany	500
5	5	Peter	USA	0
6	6	Anna	USA	0
7	7	Sam	NULL	100
8	8	Annis	UK	0

Previously, It is 15 and India

QUERY 3: UPDATE ALL THE CUSTOMERS WITH A NULL SCORE BY SETTING THEIR SCORE TO 0.

```
UPDATE customers
SET score=0
WHERE score IS NULL
```

In our table the score column has no NULL's, if there are any NULLS's it will be updated to 0.

DELETE: (Used to delete the existing rows of our table)

Always use WHERE to avoid DELETING all the rows unintentionally.

QUERY 1: DELETE ALL THE CUSTOMERS GREATER THAN ID = 5.

```
DELETE FROM customers
WHERE id>5
```

	id	first_name	country	score
1	1	Maria	Germany	350
2	2	John	USA	900
3	3	Georg	UK	750
4	4	Martin	Germany	500
5	5	Peter	USA	0

QUERY 2: DELETE ALL THE DATA FROM persons.

```
DELETE FROM persons
```

Instead of DELETE we can use **TRUNCATE**

TRUNCATE: Clears whole data from the table at once without checking or logging.(Table will be there in the Database, Data will be cleared from DB)

DELETE is slow compared to TRUNCATE.

```
TRUNCATE TABLE persons
```

TRUNCATE is way faster, it is not protocoling, it is not logging, it will just go and delete the data.