SQL intro

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL is an ANSI (American National Standards Institute) standard

---

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Some of The Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database

Operators

The following operators can be used in the WHERE clause:

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

AND / OR

SELECT & WHERE

Syntax

```
SELECT column_name, column_name
        FROM table_name;

SELECT DISTINCT column_name,column_name
        FROM table_name;

SELECT column_name,column_name
        FROM table_name
WHERE column_name operator value;
```

Example

```
SELECT * FROM students
WHERE fname=”Mohamad” AND age= 26

SELECT * FROM students
WHERE fname=”Mohamad” OR age= 26
```

SELECT * FROM students
WHERE fname=”Mohamad” OR age>26

SELECT * FROM students
WHERE fname=”Mohamad” OR age<>26

SELECT * FROM students
WHERE (id=2 OR id=4) AND age<>26

ORDER BY

Syntax
SELECT *column_name*, *column_name*
        FROM *table_name*
        ORDER BY *column_name* ASC|DESC, *column_name* ASC|DESC;

Example
SELECT * FROM students
ORDER BY age ASC // ASC är default

SELECT * FROM students
ORDER BY age DESC

SELECT * FROM students
ORDER BY age DESC, fname ASC

INSERT INTO

Syntax
INSERT INTO *table_name* (*column1*,*column2*,*column3*,...)
VALUES (*value1*,*value2*,*value3*,...);

INSERT INTO *table_name* (*column1*,*column2*,*column3*,...)
VALUES (*value1*,*value2*,*value3*,...),
          (*value1*,*value2*,*value3*,...),
          (*value1*,*value2*,*value3*,...);

Example
INSERT INTO students (fname, age)
VALUES ('Axel', 44),
          ('Henry', 64);

UPDATE

Syntax
```
UPDATE table_name
        SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

Example
```
UPDATE students
        SET fname='Chefen', age=25
WHERE id = 8

UPDATE students
        SET age=27
WHERE age = 26
```

DELETE

Syntax
```
DELETE FROM table_name
        WHERE some_column=some_value;
```

Example

```
DELETE FROM students
WHERE age < 64
```

BEGIN/ROLLBACK

UPDATE and DELETE are sensitive SQL queries. It's easy to accidentally update or delete multiple rows. Best to test the SQL query by wrapping the SQL query with BEGIN; and ROLLBACK; This makes it possible to observe the amount of affected rows without doing any changes. If the amount of affected rows are as expected, one can proceed to execute the SQL query without BEGIN/ROLLBACK

```
BEGIN;
DIN SQL QUERY;
ROLLBACK;
```