Database 101 (MySQL)

University of Stavanger

Goal

- Reviewing basic MySQL queries

- Solving tasks.

Database

- **Database (DB)**
  - Organized collection of data.
- **Database Management System (DBMS)**
  - Software that interacts with the end-users, software, and the database itself.

SQL vs NoSQL

- **SQL - Structured Query Language**
  - Table based, relational
  - **MySQL**, Oracle, SQL Server (MS), PostgreSQL, ..
- **NoSQL - Not only SQL**
  - Document based, key-value pairs, graph databases or wide-column stores.
  - MongoDB, redis, Neo4j, HBase ..

MySQL

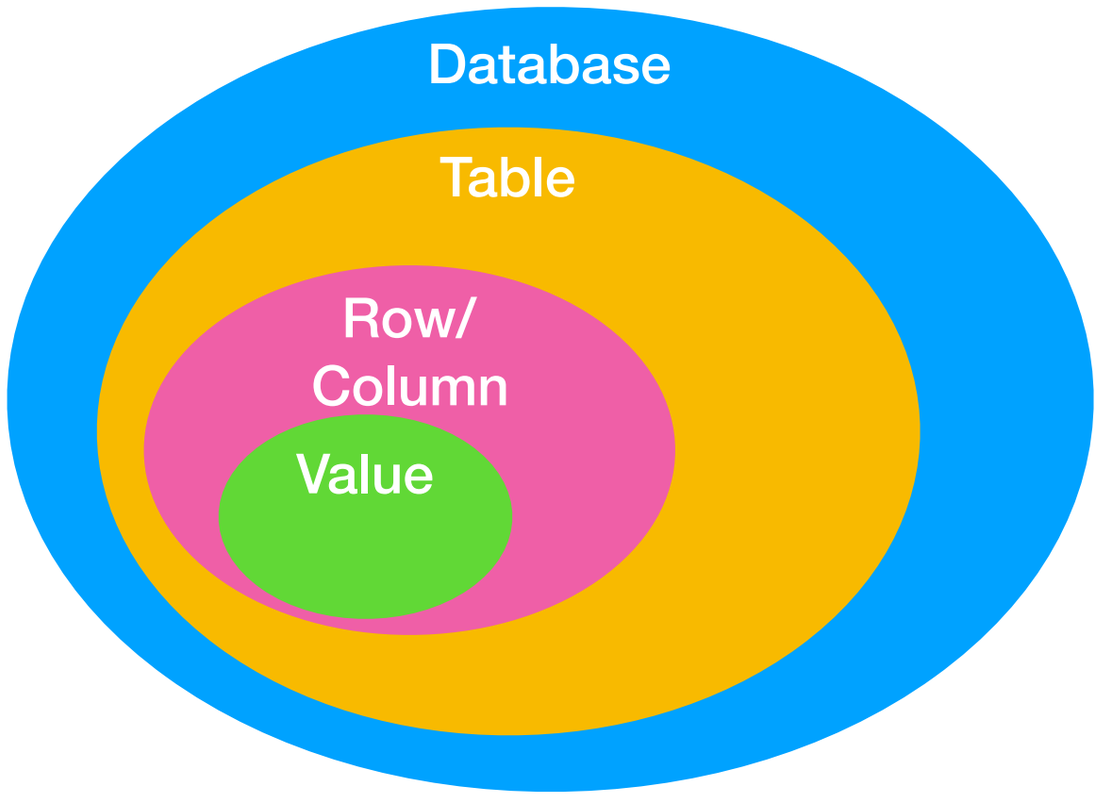
Table

idx	name	student_id	major	gpa
1	Nora	37423	physics	3,3
2	Olav	44223	biology	4,1
3	Stian	32336	law	2,9
4	Julie	99754	busines	4,3

Row

Column

Value



Creating Database/Table

- Create database and table
  - mysql> CREATE DATABASE `school`;
  - mysql> use school;
  - mysql> CREATE TABLE student (idx INTEGER, name varchar(30), student_id varchar(10), major varchar(20), gpa FLOAT);

```
mysql> CREATE DATABASE `school`;
Query OK, 1 row affected (0.00 sec)

mysql> use school;
Database changed
mysql> CREATE TABLE student ( idx INTEGER, name varchar(30), student_id varchar(10), major varchar(20), gpa FLOAT);
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
+------------------+
| Tables_in_school |
+------------------+
| student          |
+------------------+
1 row in set (0.00 sec)
```

Basic Query - INSERT

- Insert records
  - One records
    - mysql> INSERT INTO student VALUES (1, 'Nora', '37423', 'physics', 3.3);

  - Multiple records
    - mysql> INSERT INTO student VALUES
      (2, 'Olav', '44223', 'biology', 4.1),
      (3, 'Stian', '32336', 'laws', 2.9),
      (4, 'Julie', '99754', 'business', 4.3);

```
mysql> INSERT INTO student VALUES (1, 'Nora', '37423', 'physics', 3.3);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student VALUES
    -> (2, 'Olav', '44223', 'biology', 4.1),
    -> (3, 'Stian', '32336', 'laws', 2.9),
    -> (4, 'Julie', '99754', 'business', 4.3);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

WHERE

- The WHERE clause is used to filter records.
- The WHERE clause is used to extract only those records that fulfill a specified condition.
- Examples
  - SELECT * FROM student WHERE idx = 1;
  - SELECT * FROM student WHERE GPA >= 3.0;
  - UPDATE student SET major = ‘physics' WHERE major = 'biology';

Basic Query - SELECT

- Select records
  - All records
    - mysql> SELECT * FROM student;
  - Specific records
    - mysql> SELECT * FROM student WHERE name = ‘Stian';

```
mysql> SELECT * FROM student;
+------+-------+------------+---------+------+
| idx  | name  | student_id | major   | gpa  |
+------+-------+------------+---------+------+
|    1 | Nora  | 37423      | physics |  3.3 |
|    2 | Olav  | 44223      | biology |  4.1 |
|    3 | Stian | 32336      | laws    |  2.9 |
|    4 | Julie | 99754      | business |  4.3 |
+------+-------+------------+---------+------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM student WHERE name = 'Stian';
+------+-------+------------+-------+------+
| idx  | name  | student_id | major | gpa  |
+------+-------+------------+-------+------+
|    3 | Stian | 32336      | laws  |  2.9 |
+------+-------+------------+-------+------+
1 row in set (0.00 sec)
```

Basic Query - UPDATE

- Select records
  - All records
    - mysql> UPDATE student SET major = 'computer science';
  - Specific records
    - mysql> UPDATE student SET major = 'computer engineering' WHERE name =‘Nora';

```
mysql> UPDATE student SET major = 'computer science';
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> SELECT * FROM student
    -> ;
+------+-------+------------+------------------+------+
| idx  | name  | student_id | major            | gpa  |
+------+-------+------------+------------------+------+
|    1 | Nora  | 37423      | computer science |  3.3 |
|    2 | Olav  | 44223      | computer science |  4.1 |
|    3 | Stian | 32336      | computer science |  2.9 |
|    4 | Julie | 99754      | computer science |  4.3 |
+------+-------+------------+------------------+------+
4 rows in set (0.00 sec)
```

```
mysql> UPDATE student SET major = 'computer engineering' WHERE name ='Nora';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM student
    -> ;
+------+-------+------------+----------------------+------+
| idx  | name  | student_id | major                | gpa  |
+------+-------+------------+----------------------+------+
|    1 | Nora  | 37423      | computer engineering |  3.3 |
|    2 | Olav  | 44223      | computer science     |  4.1 |
|    3 | Stian | 32336      | computer science     |  2.9 |
|    4 | Julie | 99754      | computer science     |  4.3 |
+------+-------+------------+----------------------+------+
4 rows in set (0.00 sec)
```

Basic Query - DELETE

- Select records
  - All records
    - mysql> DELETE FROM student;
  - Specific records
    - mysql> DELETE FROM student WHERE idx = 4;

```
mysql> DELETE FROM student WHERE idx = 4;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM student;
+------+------+------------+----------------------+------+
| idx  | name | student_id | major                | gpa  |
+------+------+------------+----------------------+------+
|    1 | Nora | 37423      | computer engineering |  3.3 |
|    2 | Olav | 44223      | computer science     |  4.1 |
|    3 | Stian | 32336      | computer science     |  2.9 |
+------+------+------------+----------------------+------+
3 rows in set (0.00 sec)

mysql> DELETE FROM student;
Query OK, 3 rows affected (0.00 sec)

mysql> SELECT * FROM student;
Empty set (0.00 sec)
```

Aggregate Functions

- **SUM**: summation of the corresponding values.
  - mysql> SELECT SUM(gpa) FROM student;
- **AVG**: average of the corresponding values.
  - mysql> SELECT AVG(gpa) FROM student;
- **COUNT**: the number of the corresponding values.
  - mysql> SELECT count(gpa) FROM student;
- **MAX**: the maximum value of the corresponding values.
  - mysql> SELECT max(gpa) FROM student;
- **MIN** : the minimum value of the corresponding values.
  - mysql> SELECT min(gpa) FROM student;

```
mysql> SELECT SUM(gpa) FROM student;
| SUM(gpa)           |
+--------------------+
| 14.600000143051147 |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT AVG(gpa) FROM student;
+-------------------+
| AVG(gpa)          |
+-------------------+
| 3.650000035762787 |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(gpa) FROM student
+------------+
| COUNT(gpa) |
+------------+
|          4 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT MAX(gpa) FROM student;
+-------------------+
| MAX(gpa)          |
+-------------------+
| 4.300000190734863 |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT MIN(gpa) FROM student;
+--------------------+
| MIN(gpa)           |
+--------------------+
| 2.9000000953674316 |
+--------------------+
1 row in set (0.00 sec)
```

UPDATE query more

- Update values using existing values
  - UPDATE student SET **gpa = 10*gpa**;
  - UPDATE student SET student_id = **concat('uis-',student_id)**;

```
mysql> UPDATE student SET gpa = 10*gpa;
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> SELECT * FROM student;
+------+-------+------------+---------+------+
| idx  | name  | student_id | major   | gpa  |
+------+-------+------------+---------+------+
|    1 | Nora  | 37423      | physics |   33 |
|    2 | Olav  | 44223      | biology |   41 |
|    3 | Stian | 32336      | laws    |   29 |
|    4 | Julie | 99754      | business|   43 |
+------+-------+------------+---------+------+
4 rows in set (0.00 sec)
```

```
mysql> UPDATE student SET student_id = concat('uis-',student_id);
Query OK, 4 rows affected (0.01 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> SELECT * FROM student;
+------+-------+------------+---------+------+
| idx  | name  | student_id | major   | gpa  |
+------+-------+------------+---------+------+
|    1 | Nora  | uis-37423  | physics |   33 |
|    2 | Olav  | uis-44223  | biology |   41 |
|    3 | Stian | uis-32336  | laws    |   29 |
|    4 | Julie | uis-99754  | business|   43 |
+------+-------+------------+---------+------+
4 rows in set (0.00 sec)
```

ODER BY

- When you want to specify the order of your results
- e.g., Rank the student based on their GPA
  - SELECT * FROM student ORDER BY gpa **DESC**
  - DESC: descending order
  - ASC: ascending order

LIMIT

- When you want to specify the number of results. Normally, used together with ORDER BY.
- If the data is too big, it is necessary to limit the content anyway.
- Also can be used for the pagination of the contents.
    - SELECT * FROM student ORDER BY gpa DESC LIMIT 3
    - SELECT * FROM student ORDER BY gpa DESC LIMIT 3, 10
    - When its single parameter, it is the number of records.
    - When there are two parameters, first one is the starting point, and the second one is the number of the records from that starting point.

GROUP BY

- Literally to group the records based on a certain column.
- Normally used with aggregate functions.
    - What is the lowest GPA of each major?
    - What is the average height of each gender?
    - How many students are in each major?
- Examples
    - SELECT min(gpa) FROM student GROUP BY major;
    - SELECT avg(height) FROM student GROUP BY gender;
    - SELECT count(*) FROM student GROUP BY major;

HAVING

- Unlike ‘Where’, Having clause can deal with aggregate functions (e.g., sum, count, min, max).

- The majors that have more than three students

- Example

    - SELECT major FROM student HAVING count(*) > 3 GROUP BY major;

JOIN

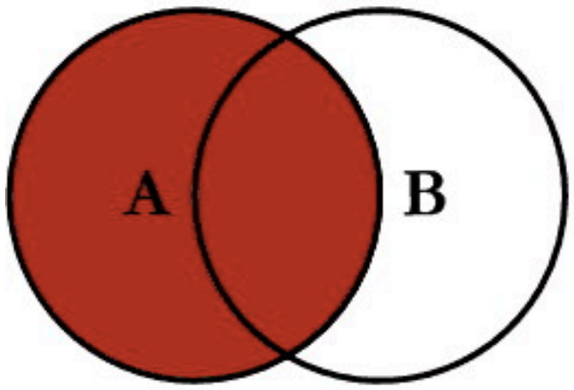
- When you have two tables for the different contents (e.g., students and major), JOIN can be used to combine these two tables.

idx	name	student_id	major	gpa
1	Nora	37423	physics	3,3
2	Olav	44223	biology	4,1
3	Stian	32336	law	2,9
4	Julie	99754	busines	4,3

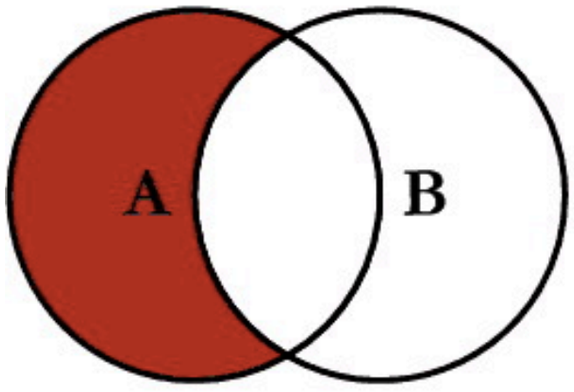
m_idx	name	location	english
1	physics	building #1	1
2	law	building #2	0
3	business	building #3	0
4	biology	building #4	1

idx	name	student_id	major	gpa	location	english
1	Nora	37423	physics	3,3	building #1	1
2	Olav	44223	biology	4,1	building #4	1
3	Stian	32336	law	2,9	building #2	0
4	Julie	99754	business	4,3	building #3	0

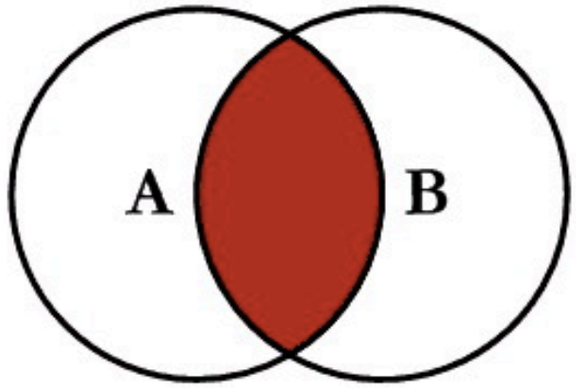
SQL JOINS



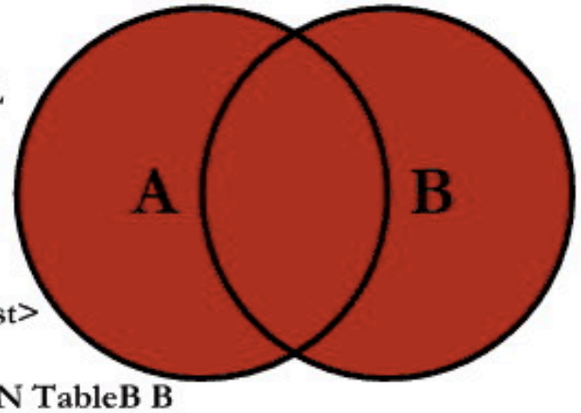

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



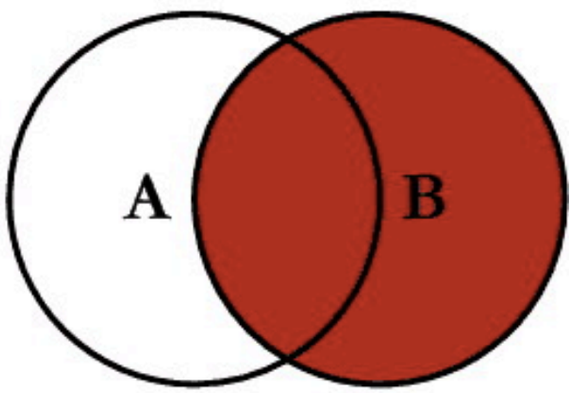

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



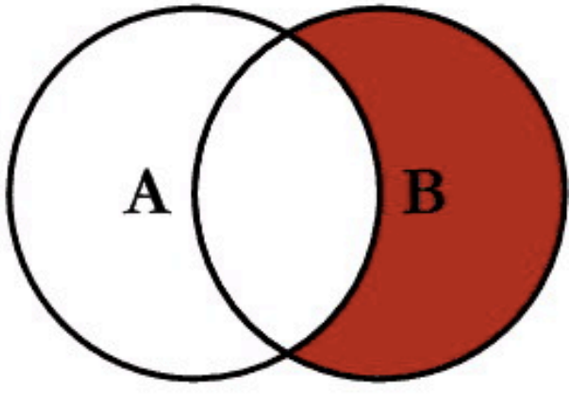

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



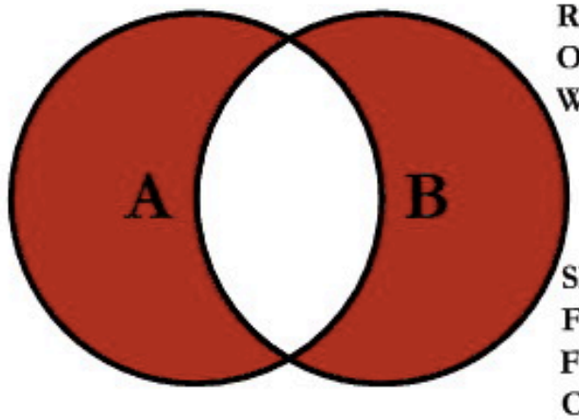

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Alias

- Can be used for table, column name, and aggregation result.
- Aliases make the query more readable.
    - (before alias) SELECT student_id FROM student WHERE student_id > 10000 ORDER BY student_id
    - (after alias) SELECT student_id **AS** s_id FROM student WHERE s_id > 10000 ORDER BY s_id
- Especially when you join tables.
    - (before alias) SELECT * FROM student JOIN major ON student.major = major.name WHERE student.gpa > 3.0 AND major.english = 1;
    - (after alias) SELECT * FROM student **AS** s JOIN major **AS** m ON s.major = m.name WHERE s.gpa > 3.0 AND m.english = 1;

Tasks