SQL for Data Science

Ruofan Chen
Ruofan.chen@uconn.edu

Mar.29 Mon 4:40pm

Contents

- SQL syntax
- Create tables
- Simple and complex queries
- Deal with different types of data
- Filtering
- Combine data

What is SQL & SQL for Data Science

SQL

- Structured Query Language is the standard language for relational database management systems and data manipulation.
- Used to query, insert, update, and modify data
- Communicate with database

SQL for Data Science

- They are *end user* of a database.
- Uses SQL to query and retrieve data, create tables.

Clarify the problem that is going to be solved

Understand organization and structure of the data

Relational Database

- Relational Database: the relationships between many tables

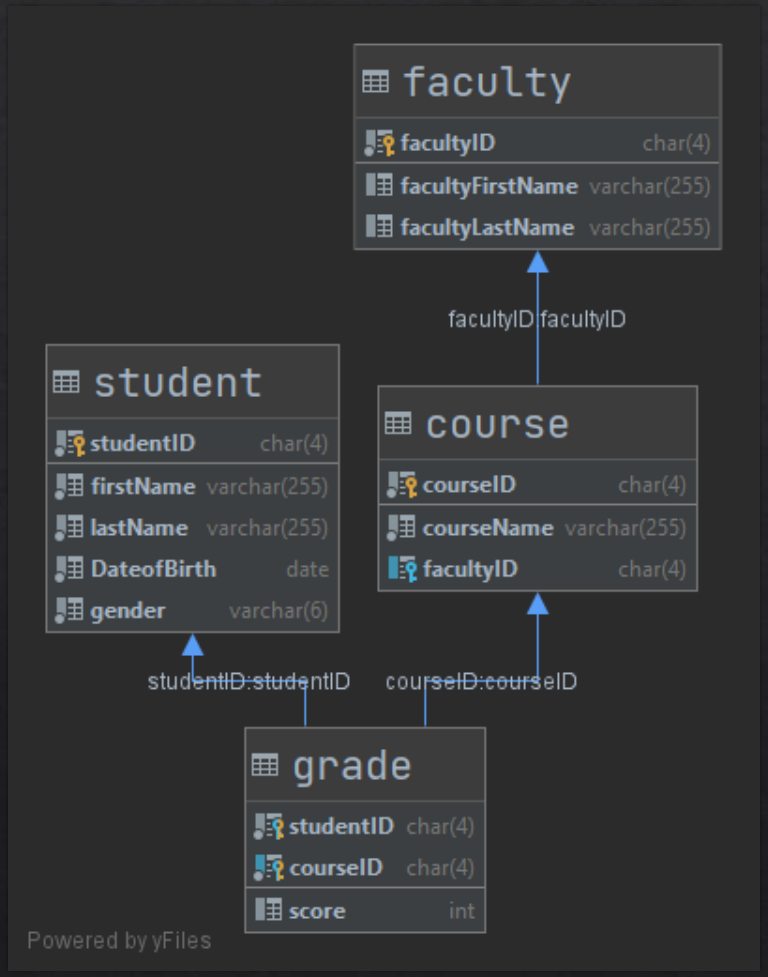
$$tables + relationships$$

- relational database
  - entity: distinguishable, unique and distinct
  - attribute:characteristic of an entity
  - relationship: association among entities

Relationships:

- one-to-one: manager to company
- one-to-many: one student to many classes
- many-to-many: students to classes

ER diagram


faculty
facultyID char(4)
facultyFirstName varchar(255)
facultyLastName varchar(255)
facultyID facultyID
student
studentID char(4)
firstName varchar(255)
lastName varchar(255)
DateofBirth date
gender varchar(6)
course
courseID char(4)
courseName varchar(255)
facultyID char(4)
studentID:studentID
courseID:courseID
grade
studentID char(4)
courseID char(4)
score int
Powered by yFiles

Creating Tables & Data Types

1. Use GUI
2. Use CREATE TABLE Statement

MySQL Data Type:

1. String Data Types
2. Numeric Data Types
3. Date and Time Data Types

String Data Types

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
TINYBLOB	For BLOBs (Binary Large OBjects). Max length: 255 bytes
BLOB(size)	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data

Numeric Data Types

Data type	Description
INT(size)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255)
TINYINT(size)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255)
SMALLINT(size)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The size parameter specifies the maximum display width (which is 255)
MEDIUMINT(size)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The size parameter specifies the maximum display width (which is 255)
BIGINT(size)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The size parameter specifies the maximum display width (which is 255)
FLOAT(size, d)	A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
DOUBLE(size, d)	A normal-size floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter
DECIMAL(size, d)	An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0.

Date and Time Data Types

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(fsp)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(fsp)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(fsp)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'

CREATE TABLE Statement

student				
studentID	firstName	lastName	DateofBirth	gender
0001	William	Patterson	1989-01-01	Male
0002	William	Patterson	1990-12-21	Female
0003	Foon Yue	Tseng	1991-12-21	Male
0004	George	Vanauf	1990-05-20	Male

```
CREATE TABLE table_name
(
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ....
);
```

```
create table student(
    studentID char(4) not null,
    firstName varchar(255) not null,
    lastName varchar(255) not null,
    DateofBirth date not null,
    gender varchar(6) not null,
    primary key (studentID)
);
```

INSERT INTO Statement

student				
studentID	firstName	lastName	DateofBirth	gender
0001	William	Patterson	1989-01-01	Male
0002	William	Patterson	1990-12-21	Female
0003	Foon Yue	Tseng	1991-12-21	Male
0004	George	Vanauf	1990-05-20	Male

INSERT INTO table_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

```
insert into student (studentid, firstName,lastName, dateofbirth,
gender)
        values
('0001','William','Patterson','1989-01-01','Male'),
('0002','William','Patterson','1990-12-21','Female'),
('0003','Foon Yue','Tseng','1991-12-21','Male'),
('0004','George','Vanauf','1990-05-20','Male');
```

Exercise

- Create following tables

course		
courseID	courseName	facultyID
0001	English	0002
0002	Math	0001
0003	Geography	0003

faculty		
facultyID	facultyFirstName	facultyLastName
0001	Larry	Bott
0002	Peter	Marsh
0003	<null>	<null>
0004		

grade		
studentID	courseID	score
0001	0001	80
0001	0002	90
0001	0003	99
0002	0002	60
0002	0003	80
0003	0001	80
0003	0002	80
0003	0003	80

SELECT Statement

SELECT column1, column2, ...
FROM table_name;

```
select firstName,lastName,gender
from student;
```

```
select firstName,lastName, gender as
stu_gen
from student;
```

SELECT * FROM table_name;

```
select *
from student;
```

SELECT DISTINCT Statement

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

```
select distinct firstName
from student;
```

Exercise

select student ID, score and calculate its percent grade

WHERE Clause

SELECT column1, column2, ...

FROM table_name

WHERE condition;

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

WHERE Clause

1

```
select studentID,score
from grade
where score<85;
```

2

```
select firstName, studentid
from student
where firstName='William';
```

3

```
select firstName,lastName,dateofbirth
from student
where dateofbirth<'1990-01-01';
```

4

```
select facultyID,facultyFirstname,facultyLastName
from faculty
where facultyFirstname is not null;
```

AND, OR and NOT Operators

AND Syntax

SELECT column1, column2, ...

FROM table_name

WHERE condition1 AND condition2 AND condition3 ...;

OR Syntax

SELECT column1, column2, ...

FROM table_name

WHERE condition1 OR condition2 OR condition3 ...;

NOT Syntax

SELECT column1, column2, ...

FROM table_name

WHERE NOT condition;

```
select lastName,firstName,gender
from student
where (firstName='William' or firstName='George')
and gender='Female';
```

BETWEEN Operator

```
SELECT column_name(s)
FROM table_name
WHERE column_name
BETWEEN value1 AND value2;
```

```
select studentID,score
from grade
where score between 60 and 90;
```

IN Operator

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

```
select firstName,gender
from student
where firstName in ('George','Foon Yue');
```

LIKE Operator

SELECT column1, column2, ...

FROM table_name

WHERE columnN LIKE pattern;

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

LIKE Operator

```
select firstName,studentid
from student
where firstName like 'G%';
```

```
select lastName,studentid
from student
where lastName like '%f';
```

COUNT() Function

COUNT Syntax

SELECT COUNT(column_name)

FROM table_name

WHERE condition;

```
select count(facultyLastName)
from faculty;
```

```
select count(*)
from faculty;
```

Exercise

find the number of distinct firstName of students

AVG() and SUM() Functions

AVG() Syntax

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

SUM() Syntax

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

MIN() and MAX() Functions

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

```
select
sum(score),
avg(score),
min(score),
max(score)
from grade;
```

Exercise

calculate total score of course 0002

GROUP BY Statement

```
SELECT column_name(s)
FROM table_name
GROUP BY column_name(s);
```

```
select gender,count(*)
from student
group by gender;
```

Exercise

list the number of males and females of students whose date of birth is after 1990-01-01

HAVING Clause

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition;
```

```
select gender,count(*)
from student
group by gender
having count(*)>2;
```

Exercise

count number of students whose name (lastName and firstName) appear in the student table more than once

ORDER BY Keyword

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition;
ORDER BY column1, column2, ...
ASC|DESC;
```

```
select *
from grade
order by score asc,
courseid desc;
```

Exercise

calculate average score of every course, sorted by avg(score) asc and course_id desc

Subqueries to filter

- Question:
- Which students have higher score than the minimum score in class 0002
- Step 1:
- Find the score of course ‘0002’
- Step 2:
- Find the student id whose score is higher than course ‘0002’ minimum score
- Step 3:
- Combine

```
select score
from grade
where courseID='0002';
```

```
select studentID,score,courseID
from grade
where score>60;
```

```
select studentID,score,courseID
from grade
where score>any(
    select score
from grade
where courseID='0002'
);
```

Subqueries to filter: ANY(), ALL() Operators

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

Subqueries to filter

Exercise

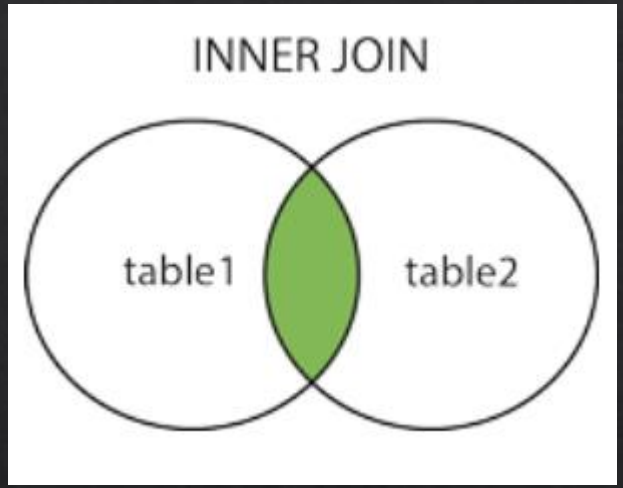
1. Which students have higher score than the maximum score in class 0002
2. find studentID and its score, and list the average score in column 3

Joining Tables

- Allows data retrieval from multiple tables in one query
- Inner join
- Left join
- Right join
- Full outer join

INNER JOIN Keyword

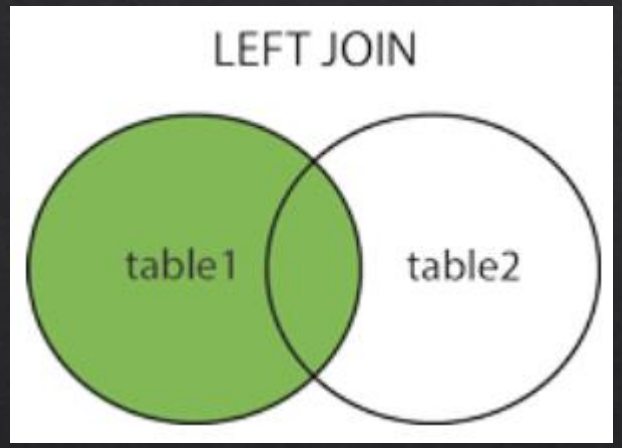
```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```




```
select a.studentID,a.firstName, a.lastName,b.courseID
from student as a inner join grade as b
on a.studentID=b.studentID;
```

LEFT JOIN Keyword

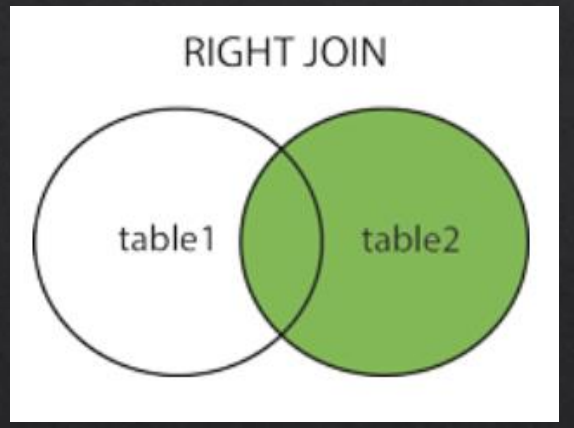
```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name =
table2.column_name;
```



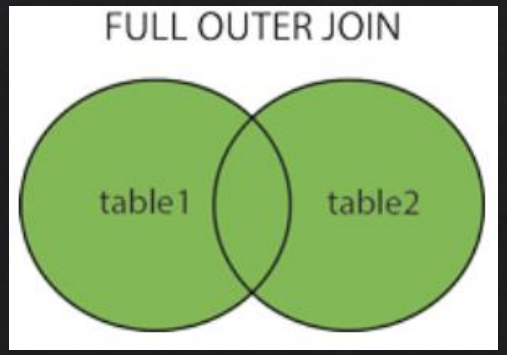

```
select a.studentID,a.lastName,a.firstName,b.courseID
from student as a left join grade as b
on a.studentID=b.studentID;
```

RIGHT JOIN and FULL OUTER JOIN Keyword

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name =
table2.column_name;
```




```
select a.studentID,a.firstName,a.lastName,b.courseID
from student as a right join grade as b
on a.studentID=b.studentID;
```



Joining Tables Exercise

list all students ID,Name,number of courses they choose,and their total score

UNION Operator

```
SELECT
column_name(s)
FROM table1

UNION

SELECT
column_name(s)
FROM table2;
```

```
select courseID,courseName
from course
union
select courseID,courseName
from course1;
```

Useful Resources and References

- SQL for Data Science (Website) https://www.coursera.org/learn/sql-for-data-science/home/welcome
- SQL tutorial (Website)

  https://www.sqltutorial.org/
- SQL tutorial by w3schools.com (Website) https://www.w3schools.com/sql/default.asp
- SQLZOO (Website)

  https://sqlzoo.net/

Related Code for this slides

- https://github.com/RuofanChen/UConn_STAT5099_21S/blob/main/0329console.sql

Thank you