

Applied Databases Relational Database Tables

HIGHER DIPLOMA IN DATA ANALYTICS





SQL

- ▶ Structured Query Language S.Q.L. “See-Quell”
- ▶ Standard Relational Database Language
- ▶ SQL is an ANSI/ISO standard, but different databases e.g. MySQL, SQL Server, Oracle may use their own proprietary extensions on top of the standard SQL.





What can SQL do?

- ▶ Create a new database
 - ▶ Create tables in a database
 - ▶ Insert data into a database
 - ▶ Read data from a database
 - ▶ Update data in a database
 - ▶ Delete data from a database
 - ▶ Manage transactions
 - ▶ Manage concurrency
 - ▶ Backup and recovery
 - ▶ Manage users
- CRUD





SQL vs MySQL

- ▶ SQL is a language.
- ▶ MySQL is a database management system.





Creating a database

- ▶ CREATE DATABASE *<database>*;

```
mysql> create database myFirstDatabase;  
Query OK, 1 row affected (0.03 sec)
```

```
mysql> CREATE dataBASE MYFirstDATABASE;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> create  
->  
-> database  
->  
-> myFirstDatabase  
->  
->  
-> ;  
Query OK, 1 row affected (0.01 sec)
```





Using a database

- SHOW DATABASES;

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| myfirstdatabase |
| mysql |
| performance_schema |
| sakila |
| sys |
| world |
+-----+
7 rows in set (0.01 sec)
```

- USE <database>;

```
mysql> use myfirstdatabase;
Database changed
```





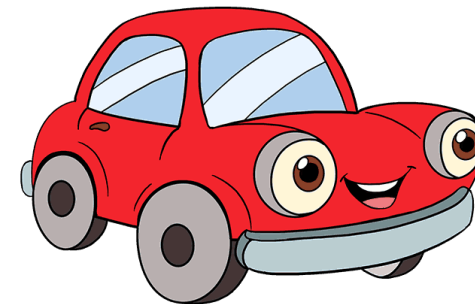
Creating Tables

► MySQL Data Types:

<https://dev.mysql.com/doc/refman/8.0/en/data-types.html>

Car Attributes

- Make
 - Model
 - Registration
 - Colour
 - Mileage
 - Engine Size
 - Cylinders
 - Crankshaft
- `varchar(20)`
 - `varchar(20)`
 - `varchar(15)`
 - `varchar(10)`
 - `integer`
 - `float(2,1)`





Creating Tables

► <https://dev.mysql.com/doc/refman/8.0/en/creating-tables.html>

► CREATE table <table> (
 <column1> <datatype>,
 <column2> <datatype>,
 <column3> <datatype>
);

```
mysql> CREATE TABLE car (  
->     make VARCHAR(20),  
->     model VARCHAR(20),  
->     registration VARCHAR(15),  
->     colour VARCHAR(10),  
->     milage INTEGER,  
->     engineSize FLOAT(2,1));  
Query OK, 0 rows affected (0.15 sec)
```



Describing Tables

- DESCRIBE <table>;

```
mysql> DESCRIBE car;
```

Field	Type	Null	Key	Default	Extra
make	varchar(20)	YES		NULL	
model	varchar(20)	YES		NULL	
registration	varchar(15)	YES		NULL	
colour	varchar(10)	YES		NULL	
milage	int(11)	YES		NULL	
engineSize	float(2,1)	YES		NULL	

```
6 rows in set (0.01 sec)
```





Creating Tables

Person Attributes

- Name
- Age
- Sex
- dob
- isStudent
- varchar(20) NOT NULL
- integer
- enum('M','F') default 'M'
- date
- tinyint(1)



```
mysql> CREATE TABLE person (  
->   name VARCHAR(20) NOT NULL,  
->   age INTEGER,  
->   sex ENUM('M','F') DEFAULT 'M',  
->   dob DATE,  
->   isStudent TINYINT(1));  
Query OK, 0 rows affected (0.11 sec)
```





Describing Tables

```
mysql> DESCRIBE person;
```

Field	Type	Null	Key	Default	Extra
name	varchar(20)	NO		NULL	
age	int(11)	YES		NULL	
sex	enum('M','F')	YES		M	
dob	date	YES		NULL	
isStudent	tinyint(1)	YES		NULL	

5 rows in set (0.00 sec)





Uniquely Identifying Rows

name	age	sex	dob	isStudent
John	23	M	2000-01-01	1
Tom	64	M	1958-03-11	0
Mary	12	F	2005-04-11	1
Alan	12	M	2005-11-21	1
Pat	29	M	1993-03-17	0
Shane	40	M	1988-07-21	0
Shane	14	M	2003-06-01	1
Alice	24	F	1999-03-01	1
Pat	37	F	1988-04-15	0



Uniquely Identifying Rows

```
mysql> DESCRIBE person;
```

Field	Type	Null	Key	Default	Extra
name	varchar(20)	NO		NULL	
age	int(11)	YES		NULL	
sex	enum('M','F')	YES		M	
dob	date	YES		NULL	
isStudent	tinyint(1)	YES		NULL	

5 rows in set (0.00 sec)

► Primary Key

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only one primary key, which may consist of single or multiple fields.





Primary Key

Person Attributes

- PersonID • integer auto_increment
- Name • varchar(20) NOT NULL
- Age • integer
- Sex • enum('M','F') default 'M'
- dob • date
- isStudent • tinyint(1)





Primary Key

```
mysql> CREATE TABLE person (  
-> personID INTEGER AUTO_INCREMENT,  
-> name VARCHAR(20) NOT NULL,  
-> age INTEGER,  
-> sex ENUM('M','F') DEFAULT 'M',  
-> dob DATE,  
-> isStudent TINYINT(1),  
-> PRIMARY KEY(personID));  
Query OK, 0 rows affected (0.17 sec)
```





Primary Key

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)





Uniquely identifying rows

- DESCRIBE <table>;

```
mysql> DESCRIBE car;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| make           | varchar(20)   | YES  |     | NULL    |       |
| model          | varchar(20)   | YES  |     | NULL    |       |
| registration   | varchar(15)   | YES  |     | NULL    |       |
| colour         | varchar(10)   | YES  |     | NULL    |       |
| milage         | int(11)       | YES  |     | NULL    |       |
| engineSize     | float(2,1)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

191-G-201

191-G-201





Primary Key

```
mysql> CREATE TABLE car (  
-> registration VARCHAR(15),  
-> make VARCHAR(20),  
-> model VARCHAR(20),  
-> colour VARCHAR(10),  
-> milage INTEGER,  
-> engineSize FLOAT(2,1),  
-> PRIMARY KEY(registration));  
Query OK, 0 rows affected (0.14 sec)
```





Getting information from a table

- ▶ SELECT <https://dev.mysql.com/doc/refman/8.0/en/select.html>

```
SELECT <columns>  
FROM <table>;
```



SELECT

```
mysql> SELECT * FROM person;
```

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT name
-> FROM person;
```

name
John
Tom
Mary
Alan
Pat
Shane
Shane
Alice
Pat

9 rows in set (0.00 sec)

```
mysql> SELECT name, age
-> FROM person;
```

name	age
John	23
Tom	64
Mary	12
Alan	12
Pat	29
Shane	40
Shane	14
Alice	24
Pat	37

9 rows in set (0.00 sec)





WHERE

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT name
-> FROM person
-> WHERE NOT isStudent;
```

name
Tom
Pat
Shane
Pat

4 rows in set (0.01 sec)

```
mysql> SELECT name
-> FROM person
-> WHERE isStudent
-> AND sex = "M";
```

name
John
Alan
Shane

3 rows in set (0.00 sec)





WHERE Operators

=	Equal To
<>	Not Equal To
!=	Not Equal To
>	Greater Than
<	Less Than
>=	Greater Than or Equal To
<=	Less Than or Equal To
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	Result is one of multiple specified values





WHERE \geq , \leq , BETWEEN

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> select personID, name, age
-> FROM person
-> WHERE age >= 20
-> AND age <= 39;
```

personID	name	age
1	John	23
5	Pat	29
8	Alice	24
9	Pat	37

4 rows in set (0.00 sec)

```
mysql> select personID, name, age
-> FROM person
-> WHERE age BETWEEN 20 and 39;
```

personID	name	age
1	John	23
5	Pat	29
8	Alice	24
9	Pat	37

4 rows in set (0.00 sec)





LIKE

- ▶ Used in a WHERE clause to search for a specified pattern in a column.
- ▶ % represents 0 or more characters

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT name, age
-> FROM person
-> WHERE name LIKE "%a%";
```

name	age
Mary	12
Alan	12
Pat	29
Shane	40
Shane	14
Alice	24
Pat	37

7 rows in set (0.01 sec)



LIKE

- _ represents a single character

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT name, age
-> FROM person
-> WHERE name LIKE "_a%";
```

name	age
Mary	12
Pat	29
Pat	37

3 rows in set (0.00 sec)





- The IN operator allows you to determine if a specified value matches any value in a set of values, or returned by a subquery.

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT personID, name
-> FROM person
-> WHERE age = 12
-> OR age = 13
-> OR age = 14
-> OR age = 15;
```

personID	name
3	Mary
4	Alan
7	Shane

3 rows in set (0.00 sec)

```
mysql> SELECT personID, name
-> FROM person
-> WHERE age IN
-> (12, 13, 14, 15);
```

personID	name
3	Mary
4	Alan
7	Shane

3 rows in set (0.00 sec)





Combining AND, OR operators

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT name, age
-> FROM person
-> WHERE sex="M"
-> AND name LIKE "S%"
-> OR name LIKE "A%"
+-----+-----+
| name | age |
+-----+-----+
| Alan | 12 |
| Shane | 40 |
| Shane | 14 |
| Alice | 24 |
+-----+-----+
4 rows in set (0.00 sec)
```

```
mysql> SELECT name, age
-> FROM person
-> WHERE sex="M"
-> AND (name LIKE "S%" OR name LIKE "A%");
+-----+-----+
| name | age |
+-----+-----+
| Alan | 12 |
| Shane | 40 |
| Shane | 14 |
+-----+-----+
3 rows in set (0.00 sec)
```

$$1 + 2 * 4 = 9$$

$$(1 + 2) * 4 = 12$$

<https://dev.mysql.com/doc/refman/8.0/en/operator-precedence.html>



LIMIT

- ▶ The LIMIT clause can be used to constrain the number of rows returned by the SELECT statement

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT name, age
-> FROM person
-> WHERE sex = "F"
-> AND age > 20;
+-----+-----+
| name  | age  |
+-----+-----+
| Alice | 24   |
| Pat   | 37   |
+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> SELECT name, age
-> FROM person
-> WHERE sex = "F"
-> AND age > 20
-> LIMIT 1;
+-----+-----+
| name  | age  |
+-----+-----+
| Alice | 24   |
+-----+-----+
1 row in set (0.00 sec)
```





LIMIT

29

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT name, age  
-> FROM person  
-> LIMIT 0,3;  
+-----+-----+  
| name | age |  
+-----+-----+  
| John | 23 |  
| Tom  | 64 |  
| Mary | 12 |  
+-----+-----+  
3 rows in set (0.00 sec)
```

```
mysql> SELECT name, age  
-> FROM person  
-> LIMIT 3,3;  
+-----+-----+  
| name | age |  
+-----+-----+  
| Alan | 12 |  
| Pat  | 29 |  
| Shane | 40 |  
+-----+-----+  
3 rows in set (0.00 sec)
```



DISTINCT



- The SELECT DISTINCT statement is used to return only distinct (different) values.

<https://dev.mysql.com/doc/refman/8.0/en/distinct-optimization.html>

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT DISTINCT(name)
-> FROM person;
```

name
John
Tom
Mary
Alan
Pat
Shane
Alice

7 rows in set (0.00 sec)



ORDER BY

► <https://dev.mysql.com/doc/refman/8.0/en/order-by-optimization.html>

```
mysql> SELECT * FROM person;
```

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT * FROM person
-> ORDER BY name;
```

personID	name	age	sex	dob	isStudent
4	Alan	12	M	2005-11-21	1
8	Alice	24	F	1999-03-01	1
1	John	23	M	2000-01-01	1
3	Mary	12	F	2005-04-11	1
5	Pat	29	M	1993-03-17	0
9	Pat	37	F	1988-04-15	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
2	Tom	64	M	1958-03-11	0

9 rows in set (0.00 sec)



ORDER BY



ASC - Ascending

DESC - Descending

YEAR() – Get Year from date

DAY() – Get Year from date

MONTH() – Get Month from date

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

```
mysql> SELECT * FROM person
-> ORDER BY name DESC, YEAR(dob);
```

personID	name	age	sex	dob	isStudent
2	Tom	64	M	1958-03-11	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
9	Pat	37	F	1988-04-15	0
5	Pat	29	M	1993-03-17	0
3	Mary	12	F	2005-04-11	1
1	John	23	M	2000-01-01	1
8	Alice	24	F	1999-03-01	1
4	Alan	12	M	2005-11-21	1

9 rows in set (0.00 sec)





Putting it all together

Name, age and Birth
Month's name

Born between 1st & 11th Show in reverse name order

```
mysql> SELECT name, age, MONTHNAME(dob)
-> FROM person
-> WHERE DAY(dob) BETWEEN 1 and 11
-> AND name NOT LIKE "A%"
-> ORDER BY name DESC;
```

name	age	MONTHNAME(dob)
Tom	64	March
Shane	14	June
Mary	12	April
John	23	January

4 rows in set (0.00 sec)

personID	name	age	sex	dob	isStudent
1	John	23	M	2000-01-01	1
2	Tom	64	M	1958-03-11	0
3	Mary	12	F	2005-04-11	1
4	Alan	12	M	2005-11-21	1
5	Pat	29	M	1993-03-17	0
6	Shane	40	M	1988-07-21	0
7	Shane	14	M	2003-06-01	1
8	Alice	24	F	1999-03-01	1
9	Pat	37	F	1988-04-15	0

9 rows in set (0.00 sec)

