

# Applied Databases MySQL Functions and Procedures

HIGHER DIPLOMA IN DATA ANALYTICS

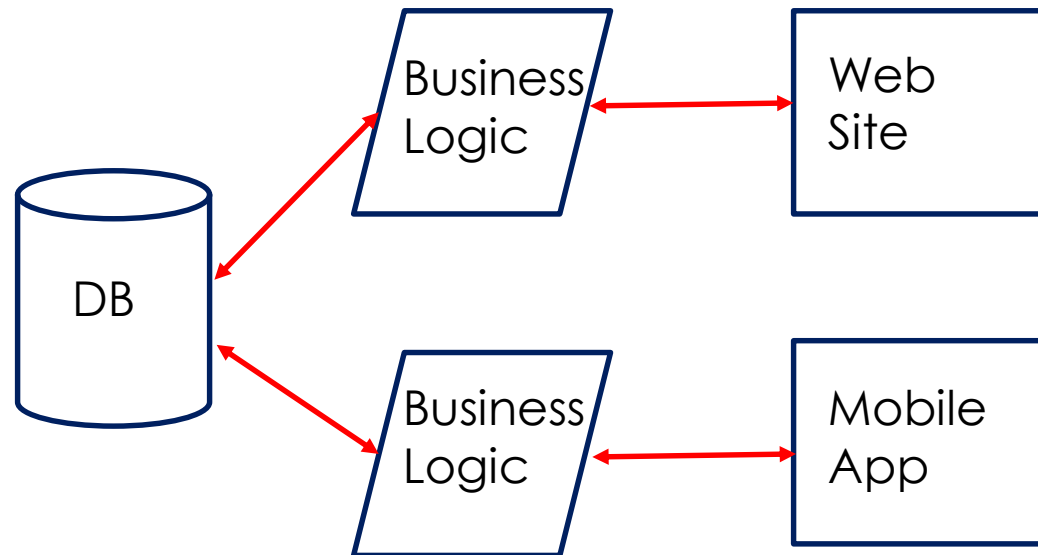


# Functions

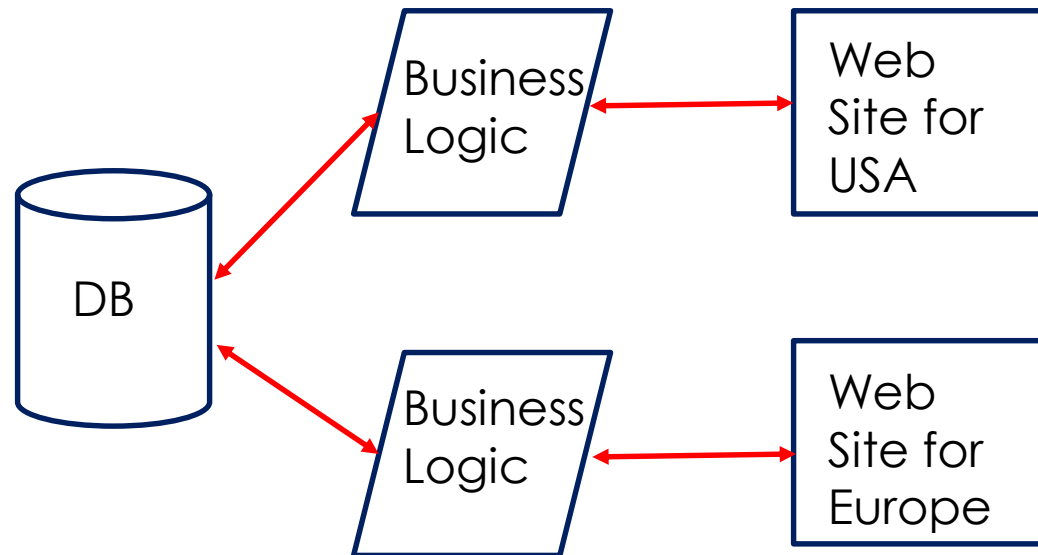
- ▶ MySQL can do more than store and retrieve data.
- ▶ It can also manipulate the data before storing or retrieving it, via functions.
- ▶ A function is a piece of code that performs some operation and returns a result.
- ▶ Some functions accept parameters, others do not.



# Why use functions



# Why use functions



# Built-in Functions

- ▶ String Functions

<https://dev.mysql.com/doc/refman/8.0/en/string-functions.html>

- ▶ Numeric Functions

<https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html>

- ▶ Date & Time Functions

<https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html>

- ▶ Aggregate Functions

<https://dev.mysql.com/doc/refman/8.0/en/group-by-functions.html>

- ▶ MySQL Information Functions

<https://dev.mysql.com/doc/refman/8.0/en/information-functions.html>

- ▶ MySQL Control Flow Functions

<https://dev.mysql.com/doc/refman/5.5/en/control-flow-functions.html>



# String Functions

- ▶ `UPPER()`
  - ▶ Returns an uppercase version of a string
- ▶ `STRCMP()`
  - ▶ Compares two strings and returns:
    - ▶ 0 if string 1 = string 2
    - ▶ -1 if string 1 < string 2
    - ▶ 1 if string 1 > string 2

```
mysql> SELECT UPPER(name)
      -> FROM subject;
+-----+
| UPPER(name) |
+-----+
| BIOLOGY     |
| COLOURING   |
| ENGLISH     |
```

```
mysql> SELECT STRCMP("MySQL", "Database");
+-----+
| STRCMP("MySQL", "Database") |
+-----+
|                               1 |
+-----+
1 row in set (0.00 sec)
```



# String Functions

- ▶ ASCII()
  - ▶ Returns the ASCII value of the first character in a string

```
mysql> select ascii("M");
+-----+
| ascii("M") |
+-----+
|          77 |
+-----+
1 row in set (0.00 sec)

mysql> select ascii("D");
+-----+
| ascii("D") |
+-----+
|          68 |
+-----+
1 row in set (0.00 sec)
```



# String Functions

- ▶ `REPLACE(string, from_string, to_string)`
  - ▶ Replaces all occurrences of a substring within a string, with a new substring.
    - ▶ *string* – The original string
    - ▶ *from\_string* – The substring to be replaced
    - ▶ *to\_string* – The new replacement string

```
mysql> SELECT tid, REPLACE(name, "Ms.", "Mrs")  
-> FROM teacher;  
+-----+-----+  
| tid | REPLACE(name, "Ms.", "Mrs") |  
+-----+-----+  
| 1 | Mr. Pasteur |  
| 2 | Mrs Dubois |  
| 3 | Mrs Smith |  
| 4 | Mr. Hawking |  
| 5 | Mr. Kavanagh |  
| 6 | Mr. Picasso |  
| 7 | Fr. Lynch |  
+-----+-----+  
7 rows in set (0.00 sec)
```





# String Functions

- ▶ SUBSTR(*string*, *start*, *length*)
  - ▶ Extract a substring from a string
    - ▶ *string* – The string to extract from
    - ▶ *start* – The start position within the string
    - ▶ *length* – The number of characters to be extracted

```
mysql> SELECT tid, SUBSTR(name, 1, 3)
-> from teacher;
```

tid	SUBSTR(name, 1, 3)
1	Mr.
2	Ms.
3	Ms.
4	Mr.
5	Mr.
6	Mr.
7	Fr.

```
7 rows in set (0.00 sec)
```



# Numeric Functions

## ► SQRT(*number*)

- Returns the square root of a number

```
mysql> SELECT SQRT(16);
+-----+
| SQRT(16) |
+-----+
|         4 |
+-----+
1 row in set (0.00 sec)
```

## ► ROUND(*number, decimals*)

- Rounds a number to a specified number of decimal places

```
mysql> SELECT engineSize
-> FROM car
-> limit 4;
+-----+
| engineSize |
+-----+
|         1.6 |
|         1.3 |
|         1.4 |
|         1.3 |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> SELECT ROUND(engineSize)
-> FROM car
-> limit 4;
+-----+
| ROUND(engineSize) |
+-----+
|                2 |
|                1 |
|                1 |
|                1 |
+-----+
4 rows in set (0.00 sec)
```



# Date Functions

- ▶ DATEDIFF(*date1*, *date2*)
  - ▶ Returns the number of days between 2 dates

```
mysql> SELECT DATEDIFF("2001-01-01", "2000-01-01");
+-----+
| DATEDIFF("2001-01-01", "2000-01-01") |
+-----+
|                                     366 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT DATEDIFF("2002-01-01", "2001-01-01");
+-----+
| DATEDIFF("2002-01-01", "2001-01-01") |
+-----+
|                                     365 |
+-----+
1 row in set (0.00 sec)
```



# Date Functions

- ▶ `DATE_FORMAT(date, format)`
  - ▶ Formats a date

```
mysql> SELECT dob
-> FROM teacher
-> where tid=5;
+-----+
| dob      |
+-----+
| 1949-11-01 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT DATE_FORMAT(dob, "%d-%m-%y")
-> FROM teacher
-> where tid=5;
+-----+
| DATE_FORMAT(dob, "%d-%m-%y") |
+-----+
| 01-11-49                      |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT DATE_FORMAT(dob, "%m/%d/%y")
-> FROM teacher
-> where tid=5;
+-----+
| DATE_FORMAT(dob, "%m/%d/%y") |
+-----+
| 11/01/49                      |
+-----+
1 row in set (0.02 sec)
```

```
mysql> SELECT dob as Birthday
-> FROM teacher
-> WHERE tid=5;
+-----+
| Birthday |
+-----+
| 1949-11-01 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT DATE_FORMAT(dob, "%m/%d/%y") as dob
-> FROM teacher
-> where tid=5;
+-----+
| dob      |
+-----+
| 11/01/49 |
+-----+
1 row in set (0.00 sec)
```

# Aggregate Functions

- ▶ An aggregate function performs a calculation on a set of values and returns a single value.

```
mysql> SELECT * FROM teacher;
```

tid	Name	level	experience	dob
1	Mr. Pasteur	L	15	1960-02-02
2	Ms. Dubois	L	22	1967-09-02
3	Ms. Smith	J	4	1980-03-23
4	Mr. Hawking	L	40	1951-02-19
5	Mr. Kavanagh	J	50	1949-11-01
6	Mr. Picasso	J	42	1939-03-30
7	Fr. Lynch	L	55	1939-03-31

```
7 rows in set (0.00 sec)
```

```
mysql> SELECT AVG(experience)
-> FROM teacher;
```

AVG(experience)
32.5714

```
1 row in set (0.00 sec)
```

- ▶ AVG()



# Aggregate Functions

- ▶ MIN(), MAX(), SUM(), COUNT()

```
mysql> SELECT COUNT(*)  
      -> FROM teacher  
      -> WHERE level="L";  
+-----+  
| COUNT(*) |  
+-----+  
|         4 |  
+-----+  
1 row in set (0.00 sec)
```



# Aggregate Functions – Group By

- The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
mysql> SELECT * FROM teacher;
```

tid	Name	level	experience	dob
1	Mr. Pasteur	L	15	1960-02-02
2	Ms. Dubois	L	22	1967-09-02
3	Ms. Smith	J	4	1980-03-23
4	Mr. Hawking	L	40	1951-02-19
5	Mr. Kavanagh	J	50	1949-11-01
6	Mr. Picasso	J	42	1939-03-30
7	Fr. Lynch	L	55	1939-03-31

```
7 rows in set (0.00 sec)
```

```
mysql> SELECT AVG(experience)
-> FROM teacher;
```

AVG(experience)
32.5714

```
1 row in set (0.00 sec)
```

```
mysql> SELECT level, AVG(experience)
-> FROM teacher
-> GROUP BY level;
```

level	AVG(experience)
J	32.0000
L	33.0000

```
2 rows in set (0.00 sec)
```



# Aggregate Functions – Group By

```
mysql> select * from car;
```

registration	make	model	colour	milage	engineSize
05-M0-17931	Toyota	Highlander	Green	253789	1.6
10-G-2334	Toyota	Corolla	Green	123389	1.3
10-WH-17931	Toyota	Corolla	Silver	130389	1.4
11-M0-23431	Toyota	Corolla	Black	1234123	1.3
12-WH-123	Ford	Ka	Black	125882	1.0
132-G-9923	Ford	Ka	Silver	325883	1.0
132-M0-19323	Ford	Galaxy	Silver	2343	1.5
171-G-39532	Toyota	Corolla	Silver	55882	1.3
171-M0-12533	Ford	Fiesta	Black	25882	1.0
99-G-300	Toyota	Corolla	Green	599339	1.3

```
10 rows in set (0.00 sec)
```

```
mysql> SELECT ROUND(AVG(milage)) as "KMs"
-> FROM car;
```

```
+-----+
| KMs    |
+-----+
| 287690 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT model, ROUND(AVG(milage)) as "KMs"
-> FROM car
-> GROUP BY model;
```

```
+-----+-----+
| model  | KMs    |
+-----+-----+
| Corolla | 428624 |
| Fiesta  | 25882  |
| Galaxy  | 2343   |
| Highlander | 253789 |
| Ka      | 225883 |
+-----+-----+
```

```
5 rows in set (0.00 sec)
```





# Aggregate Functions – Group By

```
mysql> select * from car;
```

registration	make	model	colour	milage	engineSize
05-M0-17931	Toyota	Highlander	Green	253789	1.6
10-G-2334	Toyota	Corolla	Green	123389	1.3
10-WH-17931	Toyota	Corolla	Silver	130389	1.4
11-M0-23431	Toyota	Corolla	Black	1234123	1.3
12-WH-123	Ford	Ka	Black	125882	1.0
132-G-9923	Ford	Ka	Silver	325883	1.0
132-M0-19323	Ford	Galaxy	Silver	2343	1.5
171-G-39532	Toyota	Corolla	Silver	55882	1.3
171-M0-12533	Ford	Fiesta	Black	25882	1.0
99-G-300	Toyota	Corolla	Green	599339	1.3

```
10 rows in set (0.00 sec)
```

```
mysql> SELECT model, ROUND(AVG(milage)) as "KMs"
-> FROM car
-> WHERE milage > 60000
-> GROUP BY model;
```

model	KMs
Corolla	521810
Highlander	253789
Ka	225883

```
3 rows in set (0.00 sec)
```



# Aggregate Functions – Group By

- ▶ The HAVING clause is often used with the GROUP BY clause to filter groups based on a specified condition.
- ▶ If the GROUP BY clause is omitted, the HAVING clause behaves like the WHERE clause.
- ▶ The HAVING clause applies a filter condition to each group of rows.
- ▶ The WHERE clause applies the filter condition to each individual row.

```
mysql> SELECT model, ROUND(AVG(milage)) as "KMs"  
-> FROM car  
-> WHERE milage > 60000  
-> GROUP BY model;  
+-----+-----+  
| model      | KMs      |  
+-----+-----+  
| Corolla    | 521810    |  
| Highlander | 253789    |  
| Ka         | 225883    |  
+-----+-----+  
3 rows in set (0.00 sec)
```

```
mysql> SELECT model, ROUND(AVG(milage)) as "KMs"  
-> FROM car  
-> WHERE milage > 60000  
-> GROUP BY model  
-> HAVING KMs > 250000;  
+-----+-----+  
| model      | KMs      |  
+-----+-----+  
| Corolla    | 521810    |  
| Highlander | 253789    |  
+-----+-----+  
2 rows in set (0.00 sec)
```



# MySQL Information Functions

- ▶ MySQL provides some information functions such as

- ▶ DATABASE()

```
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| lab1       |
+-----+
1 row in set (0.00 sec)
```

- ▶ USER()

```
mysql> SELECT USER();
+-----+
| USER()      |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)
```



# MySQL Control Flow Functions

- ▶ `IF(condition, value_if_true, value_if_false)`
  - ▶ *condition* - Value to Test
  - ▶ *value\_if\_true* – Value to return if *condition* is True
  - ▶ *value\_if\_false* – Value to return if *condition* is False

```
mysql> SELECT IF(150>200,"Yes", "No") "T/F";
+-----+
| T/F |
+-----+
| No  |
+-----+
1 row in set (0.00 sec)
```



# MySQL Control Flow Functions

```
mysql> select * from teacher;
```

tid	Name	level	experience	dob
1	Mr. Pasteur	L	15	1960-02-02
2	Ms. Dubois	L	22	1967-09-02
3	Ms. Smith	J	4	1980-03-23
4	Mr. Hawking	L	40	1951-02-19
5	Mr. Kavanagh	J	50	1949-11-01
6	Mr. Picasso	J	42	1939-03-30
7	Fr. Lynch	L	55	1939-03-31

```
7 rows in set (0.00 sec)
```

```
mysql> SELECT *, IF(experience >= 20 AND experience <=45, "Y", "") as "Payrise Due"
-> from teacher;
```

tid	Name	level	experience	dob	Payrise Due
1	Mr. Pasteur	L	15	1960-02-02	
2	Ms. Dubois	L	22	1967-09-02	Y
3	Ms. Smith	J	4	1980-03-23	
4	Mr. Hawking	L	40	1951-02-19	Y
5	Mr. Kavanagh	J	50	1949-11-01	
6	Mr. Picasso	J	42	1939-03-30	Y
7	Fr. Lynch	L	55	1939-03-31	

```
7 rows in set (0.00 sec)
```



# MySQL Control Flow Functions

- ▶ CASE WHEN *condition 1* THEN *result 1*  
    WHEN *condition 2* THEN *result 1*  
    WHEN *condition n* THEN *result n*  
    ELSE *result*  
END



# MySQL Control Flow Functions

```
mysql> select name, dob
-> from person;
```

name	dob
John	2000-01-01
Tom	1958-03-11
Mary	2005-04-11
Alan	2005-11-21
Pat	1993-03-17
Shane	1988-07-21
Shane	2003-06-01
Alice	1999-03-01
Pat	1988-04-15

9 rows in set (0.00 sec)

```
mysql> select name, dob,
-> CASE
-> WHEN month(dob) in (2,3,4) THEN "Spring"
-> WHEN month(dob) in (5,6,7) THEN "Summer"
-> WHEN month(dob) in (8,9,10) THEN "Autumn"
-> WHEN month(dob) in (11,12,1) THEN "Winter"
-> END as Season
-> from person;
```

name	dob	Season
John	2000-01-01	Winter
Tom	1958-03-11	Spring
Mary	2005-04-11	Spring
Alan	2005-11-21	Winter
Pat	1993-03-17	Spring
Shane	1988-07-21	Summer
Shane	2003-06-01	Summer
Alice	1999-03-01	Spring
Pat	1988-04-15	Spring

9 rows in set (0.00 sec)



# MySQL Control Flow Functions

```
mysql> select name, dob
-> from person;
```

name	dob
John	2000-01-01
Tom	1958-03-11
Mary	2005-04-11
Alan	2005-11-21
Pat	1993-03-17
Shane	1988-07-21
Shane	2003-06-01
Alice	1999-03-01
Pat	1988-04-15

9 rows in set (0.00 sec)

```
mysql> select name, dob,
-> CASE
-> WHEN month(dob) in (2,3,4) THEN "Spring"
-> WHEN month(dob) in (5,6,7) THEN "Summer"
-> ELSE ""
-> END as Season
-> from person;
```

name	dob	Season
John	2000-01-01	
Tom	1958-03-11	Spring
Mary	2005-04-11	Spring
Alan	2005-11-21	
Pat	1993-03-17	Spring
Shane	1988-07-21	Summer
Shane	2003-06-01	Summer
Alice	1999-03-01	Spring
Pat	1988-04-15	Spring

9 rows in set (0.00 sec)





# MySQL Stored Routines

- ▶ A Stored Routine is user-written code that extends the functionality of MySQL.

## Uses

- ▶ When multiple client applications are written in different languages or work on different platforms, but need to perform the same database operations.
- ▶ To ensure security. Applications cannot directly access tables only stored routines.



# MySQL Stored Routines

## ► Advantages

### ► Speed

Performance of applications accessing the database is increased.

This is because stored procedures are compiled and stored in the database.

### ► Traffic

Instead of sending multiple lengthy SQL statements, the application has to send only the name and parameters of the stored routine.

## ► Disadvantages

### ► Complexity

Not designed for complex business logic.

### ► Difficult to debug

Only a few database management systems allow you to debug stored procedures. MySQL is not one of them.

### ► Performance

A DBMS is not well-designed for logical operations.



# MySQL Stored Functions


- ▶ A stored function is a special kind stored routine that returns a single value.
- ▶ Stored functions are used to encapsulate common formulas or business rules that are reusable among SQL statements or stored routines.
- ▶ Functions take 0 or more input parameters and return a single value.

<https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html>



# MySQL Stored Functions

```
CREATE FUNCTION add2Nums (num1 integer, num2 integer)
RETURNS integer
DETERMINISTIC
BEGIN
    RETURN num1 + num2;
END
```



```
mysql> SELECT add2Nums(3,10);
+-----+
| add2Nums(3,10) |
+-----+
|           13 |
+-----+
1 row in set (0.00 sec)
```



# MySQL Stored Functions

```
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 discount(14);
```

discount(14)
0%

```
1 row in set (0.00 sec)
```

```
mysql> select discount(74);
```

discount(74)
40%

```
1 row in set (0.00 sec)
```

```
CREATE FUNCTION discount(age INT(11))
RETURNS VARCHAR(3)
DETERMINISTIC
BEGIN
    IF age < 16 THEN
        RETURN "0%";
    ELSEIF age < 26 THEN
        RETURN "10%";
    ELSEIF age < 40 THEN
        RETURN "20%";
    ELSEIF age < 60 THEN
        RETURN "30%";
    ELSE
        RETURN "40%";
    END IF;
END
```



# MySQL Stored Functions

```
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)
```

```
CREATE FUNCTION discount(age INT(11))
RETURNS VARCHAR(3)
DETERMINISTIC
BEGIN
  IF age < 16 THEN
    RETURN "0%";
  ELSEIF age < 26 THEN
    RETURN "10%";
  ELSEIF age < 40 THEN
    RETURN "20%";
  ELSEIF age < 60 THEN
    RETURN "30%";
  ELSE
    RETURN "40%";
  END IF;
END
```

```
mysql> SELECT name, age, discount(age) "Discount"
-> FROM person;
```

name	age	Discount
John	23	10%
Tom	64	40%
Mary	12	0%
Alan	12	0%
Pat	29	20%
Shane	40	30%
Shane	14	0%
Alice	24	10%
Pat	37	20%

```
9 rows in set (0.00 sec)
```



# MySQL Stored Procedures

FUNCTIONS	PROCEDURES
Return a single value	Return 0 or more values
Only SELECT	SELECT, INSERT, UPDATE, DELETE
Can't use Stored Procedures	Can use Stored Functions
Does not support Transactions	Supports Transactions

<https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html>



# MySQL Stored Procedures

```
mysql> select * from car;
```

registration	make	model	colour	milage	engineSize
05-M0-17931	Toyota	Highlander	Green	253789	1.6
10-G-2334	Toyota	Corolla	Green	123389	1.3
10-WH-17931	Toyota	Corolla	Silver	130389	1.4
11-M0-23431	Toyota	Corolla	Black	1234123	1.3
12-WH-123	Ford	Ka	Black	125882	1.0
132-G-9923	Ford	Ka	Silver	325883	1.0
132-M0-19323	Ford	Galaxy	Silver	2343	1.5
171-G-39532	Toyota	Corolla	Silver	55882	1.3
171-M0-12533	Ford	Fiesta	Black	25882	1.0
99-G-300	Toyota	Corolla	Green	599339	1.3

```
10 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM car
-> WHERE make = "Toyota"
-> AND milage < 200000
-> ORDER BY milage;
```

registration	make	model	colour	milage	engineSize
171-G-39532	Toyota	Corolla	Silver	55882	1.3
10-G-2334	Toyota	Corolla	Green	123389	1.3
10-WH-17931	Toyota	Corolla	Silver	130389	1.4

```
3 rows in set (0.00 sec)
```



# MySQL Stored Procedures

```
mysql> select * from car;
```

registration	make	model	colour	milage	engineSize
05-M0-17931	Toyota	Highlander	Green	253789	1.6
10-G-2334	Toyota	Corolla	Green	123389	1.3
10-WH-17931	Toyota	Corolla	Silver	130389	1.4
11-M0-23431	Toyota	Corolla	Black	1234123	1.3
12-WH-123	Ford	Ka	Black	125882	1.0
132-G-9923	Ford	Ka	Silver	325883	1.0
132-M0-19323	Ford	Galaxy	Silver	2343	1.5
171-G-39532	Toyota	Corolla	Silver	55882	1.3
171-M0-12533	Ford	Fiesta	Black	25882	1.0
99-G-300	Toyota	Corolla	Green	599339	1.3

10 rows in set (0.00 sec)

```
CREATE PROCEDURE make_milage(mk VARCHAR(20), ml INT(11))
DETERMINISTIC
BEGIN
    SELECT * FROM CAR
    WHERE make LIKE mk
    AND milage < ml
    ORDER BY milage;
END
```

```
mysql> call make_milage("Toyota", 200000);
```

registration	make	model	colour	milage	engineSize
171-G-39532	Toyota	Corolla	Silver	55882	1.3
10-G-2334	Toyota	Corolla	Green	123389	1.3
10-WH-17931	Toyota	Corolla	Silver	130389	1.4

3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

```
mysql> call make_milage("", 60000);
```

registration	make	model	colour	milage	engineSize
132-M0-19323	Ford	Galaxy	Silver	2343	1.5
171-M0-12533	Ford	Fiesta	Black	25882	1.0
171-G-39532	Toyota	Corolla	Silver	55882	1.3

3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

```
mysql> call make_milage("Ford", 5000);
```

registration	make	model	colour	milage	engineSize
132-M0-19323	Ford	Galaxy	Silver	2343	1.5

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

# MySQL Stored Routine Management

- ▶ Finding Functions and Procedures

```
mysql> select routine_name, routine_type from information_schema.routines
-> where routine_name IN ("add2nums", "discount", "make_milage");
+-----+-----+
| ROUTINE_NAME | ROUTINE_TYPE |
+-----+-----+
| add2nums     | FUNCTION     |
| discount     | FUNCTION     |
| make_milage  | PROCEDURE    |
+-----+-----+
3 rows in set (0.00 sec)
```

- ▶ What's in a Function or Procedure

```
mysql> SHOW CREATE FUNCTION add2nums;
```

- ▶ Delete a Function or Procedure

```
mysql> DROP FUNCTION add2nums;
Query OK, 0 rows affected (0.00 sec)
```

