Date & Time Functions

CURDATE() CURTIME() NOW()	Returns the current date, time, or date and time in 'YYYY-MM-DD', 'HH:MM:SS', or 'YYYY-MM-DD HH:MM:SS' format	SELECT CURDATE(); returns '2019-01-25' SELECT CURTIME(); returns '21:05:44' SELECT NOW(); returns '2019-01-25 21:05:44'
DATE(expr) TIME(expr)	Extracts the date or time from a date or datetime expression expr	SELECT DATE('2013-03-25 22:11:45'); returns '2013-03-25' SELECT TIME('2013-03-25 22:11:45'); returns '22:11:45'
DAY(d) MONTH(d) YEAR(d)	Returns the day, month, or year from date d	SELECT DAY('2016-10-25'); returns 25 SELECT MONTH('2016-10-25'); returns 10 SELECT YEAR('2016-10-25'); returns 2016
HOUR(t) MINUTE(t) SECOND(t)	Returns the hour, minute, or second from time t	SELECT HOUR('22:11:45'); returns 22 SELECT MINUTE('2-2:11:45'); returns 11 SELECT SECOND('22:11:45'); returns 45
DATEDIFF(expr1, expr2) TIMEDIFF(-expr1, expr2)	Returns expr1 - expr2 in number of days or time values, given expr1 and expr2 are date, time, or datetime values	SELECT DATEDIFF('2013-03-10', '2013-03-04'); returns 6 SELECT TIMEDIFF('10:00:00', '09:45:30'); returns 00:14:30

Aggregate Functions

COUNT()	Count number of rows in the set.	SELECT COUNT(*) FROM Employee WHERE Bonus > 500;
MIN()	Find minimum value in set.	SELECT MIN(Salary) FROM Employee;
MAX()	Find max value in set.	SELECT MAX(Salary) FROM Employee;
SUM()	Sum all values in set.	SELECT SUM(Salary) FROM Employee;
AVG()	Compute mean of all values in set.	SELECT AVG(Salary) FROM Employee;

An aggregate function processes values from a set of rows and returns a summary value. They appear in a SELECT clause and process all rows that satisfy the WHERE clause condition. If a SELECT statement has no WHERE clause, the aggregate function processes all rows.

Ignores NULL values.

Numeric Functions

ABS(n)	Returns the absolute value of n	SELECT ABS(-5); returns 5

Numeric Functions (cont)		
LOG(n	Returns the natural logarithm of n	SELECT LOG(10); returns 2.30258509299404
POW(x, y)	Returns x to the power of y	SELECT POW(2, 3); returns 8
RAND()	Returns a random number between 0 (inclusive) and 1 (exclusive	SELECT RAND(); returns 0.1183182570322586
ROUND(n, d)	Returns n rounded to d decimal places	SELECT ROUND(16.25, 1); returns 16.3
SQRT(n)	Returns the square root of n	SELECT SQRT(25); returns 5

String Functions		
CONCAT(s1, s2, ...)	Returns the string that results from concatenating the string arguments	SELECT CONCAT('Dis', 'en', 'gage'); returns 'Disengage'
LOWER(s)	Returns the lowercase s	SELECT LOWER('MySQL'); returns 'mysql'
REPLACE(s, from, to)	Returns the string s with all occurrences of from replaced with to	SELECT REPLACE('This and that', 'and', 'or'); returns 'This or that'
SUBSTRING(s, pos, len)	Returns the substring from s that starts at position pos and has length len	SELECT SUBSTRING('Boomerang', 1, 4); returns 'Boom'
TRIM(s)	Returns the string s without leading and trailing spaces	SELECT TRIM(' test '); returns 'test'
UPPER(s)	Returns the uppercase s	SELECT UPPER('mysql'); returns 'MYSQL'

SQL Sublanguages

DDL	Data Definition Language	Defines DB structure.
DQL	Data Query Language	Retrieve data from DB.
DML	Data Manipulation Language	Manipulate data stored in DB.
DCL	Data Control Language	Control DB user access.
DTL	Data Transaction Language	Manage DB transactions.

Comments

```
-- single line comment
/* multi-line
      Comment */
```

Literals

```
'String'
" Str ing "
123
x'0fa2'
```

[S]ingle quotes are for [S]trings Literals (date literals are also strings);
[D]ouble quotes are for [D]atabase Identifiers;
Explicit values that are string, numeric, or binary.
Strings must be surrounded by single quotes or double quotes.
Binary values are represented with x'0' where the 0 is any hex value.

USE & SHOW

USE DatabaseName	Select default db for use.	Use World;
SHOW DATABASES	lists all databases in the database system instance.	SHOW DATABASES;
SHOW TABLES	lists all tables in the default database.	SHOW TABLES;

USE & SHOW (cont)

SHOW COLUMNS FROM TableName	lists all columns in the TableName table of the default database.	SHOW COLUMNS FROM CountryLanguage;
SHOW CREATE TABLE TableName	shows the CREATE TABLE statement for the TableName table of the default database.	

Additional SHOW statements generate information about system errors, configuration, privileges, logs, etc.

Create & Drop Databases

```
CREATE DATABASE petStore;
DROP DATABASE petStore;
```

Create & Drop Table

```
CREATE TABLE Horse (
       ID SMALLINT UNSIGNED AUTO_I NCR EMENT,
       Name VARCHA R(15) NOT NULL DEFAULT 'Sam',
        BREED VARCHA R(20) CHECK (Breed = 'Quarter
Horse' OR 'Saddl ebr ed'),
        PRIMARY KEY (ID)
);
DROP TABLE Horse;
```

INSERT

```
INSERT INTO Product (ID, Name, ProductType,
OriginDate, Weight) VALUES
    (100, 'Trico rder', 'TC', '2020- 08-11', 2.4),
    (200, 'Food replic ator', 'FOD', '2020- 09-
21', 54.2),
    (300, 'Cloaking device', 'CD', '2019- 02-04',
177.9);
```

MySQL Data Types

CHAR	String (0-255)
VARCHAR	String (0-255
TINYINT	-128 to 127
SMALLINT	-32768 to 32767
MEDIUMINT	-8388608 to 8388607
INT	-2147483648 to 8388607
BIGINT	-9223372036854775808 to 9223372036854775807
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MM:SS
DECIMAL(M,D)	Numeric values with M digits, of which D digits follow the decimal point

UNSIGNED Integers have the same range but start from zero.

Arithmetic Operators

Returns NULL when either operand is NULL

ALTER TABLE

ADD	Add column	ALTER TABLE TableName ADD ColumnName DataType;
CHANGE	Modify column	ALTER TABLE TableName CHANGE CurrentColumnName NewColumnName NewDataType;
DROP	Delete column	ALTER TABLE TableName DROP ColumnName;

Create View

```
CREATE VIEW ViewName [ ( Column1, Column2, ... ) ]
AS Select Sta tement;
CREATE VIEW Manage rView
AS SELECT Depart men tName, Employ eeName AS
Manage rName
     FROM Depart ment, Employee
      WHERE ManagerID = Employ eeID;
CREATE VIEW ViewName [ ( Column1, Column2, ... ) ]
```

Create View (cont)

```
AS Select Sta tement
[ WITH CHECK OPTION ];
```

When WITH CHECK OPTION is specified, the database rejects inserts and updates that do not satisfy the view query WHERE clause.

DISTINCT

```
SELECT DISTINCT Language
FROM Countr yLa nguage
WHERE IsOfficial = 'F';
```

Unique values.

WHERE IN

```
SELECT *
FROM Countr yLa nguage
WHERE Language IN ('Dutch', 'Kongo', 'Alban ian');
```

Determine if a value matches one of several values.

BETWEEN

```
SELECT Name
FROM Employee
WHERE HireDate BETWEEN '2000- 01-01' AND '2020- -
01-01';
```

Value BETWEEN minValue AND maxValue and is equivalent to value >= minValue AND value <= maxValue.

LIKE

```
SELECT *
FROM Countr yLa nguage
WHERE Countr yCode LIKE 'A_W';
```

% matches any number of characters. Ex: LIKE 'L%t' matches "Lt", "Lot", "Lift", and "Lol cat".
_ matches exactly one character. Ex: LIKE 'L_t' matches "Lot" and "-Lit" but not "Lt" and "Loot".

Case-insensitive by default Case-sensitive if followed by the BINARY keyword. Ex: LIKE BINARY 'L%t' matches 'Left' but not 'left'.

Wildcard search % or _, a backslash (\) must precede % or _. Ex: LIKE 'a\%' matches "a%".

ORDER BY

```
-- Order by Language (ascending)
SELECT *
FROM Countr yLa nguage
ORDER BY Language;
-- Order by Language (desce nding)
SELECT *
FROM Countr yLa nguage
ORDER BY Language DESC;
-- Order by Countr yCode, then Language
SELECT *
FROM Countr yLa nguage
ORDER BY Countr yCode, Language;
```

Order selected rows by one or more columns in ascending order. DESC orders rows in descending order.

GROUP BY

```
SELECT CountryCode, SUM(Population)
FROM City
GROUP BY Countr yCode;
SELECT Countr yCode, District, COUNT(*)
FROM City
GROUP BY Countr yCode, District;
```

Commonly used with aggregate functions. GROUP BY and one or more columns. Each simple or composite value of the column(s) becomes a group. The query computes the aggregate function separately, and returns one row, for each group.

Appears between the WHERE clause, if any, and the ORDER BY clause.

HAVING

```
SELECT CountryCode, SUM(Population)
FROM City
GROUP BY Countr yCode
HAVING SUM(Po pul ation) > 2300000;
SELECT Countr yCode, District, COUNT(*)
FROM City
GROUP BY Countr yCode, District
HAVING COUNT(*) >= 2;
```

Used with GROUP By to filter group results. Follows GROUP BY and precedes ORDER BY.

Prefix & Alias

```
SELECT Department.Name AS Group,
               Emp loy ee.Name AS Supervisor
FROM Depart ment, Employee
WHERE Manager = ID;
```

Prefix is the TableName.ColumnName.
Alias = AS
The AS keyword is optional and may be omitted. Ex: SELECT Name N FROM Country C.

Join Query

```
SELECT DepartmentName, EmployeeName
FROM Depart ment, Employee
WHERE Manager = ID;
SELECT Depart men t.Name AS Group,
               Emp loy ee.Name AS Supervisor
FROM Department
INNER JOIN Employee
ON Manager = ID;
```

SELECT LeftColumn, RightColumn
FROM LeftTable, RightTable
WHERE Key = Key;

FROM specifies the left table.
INNER JOIN or FULL JOIN specifies the right table.
ON specifies the join columns.

Join Types

INNER JOIN	Default. Only matching left and right table rows.
FULL (OUTER) JOIN	Many DB do not support. All left and right table rows, regardless of match.
LEFT(OUTER) JOIN	All left table rows, only matching right table rows.
RIGHT (OUTER) JOIN	Many DB do not support. All right table rows, only matching left table rows.

Union Full Join

```
SELECT * FROM Table1
LEFT OUTER JOIN Table2
ON Table1.co lum n_m atc h=T abl e2.c ol umn _match
UNION
SELECT * FROM Table1
RIGHT OUTER JOIN Table2
ON Table1.co lum n_m atc h=T abl e2.c ol umn -
_match;
```

Use for FULL JOINS in MtSQL. Similar to JOIN but JOIN is good practice. The first SELECT returns matching rows and the second SELECT returns unmatched Department rows. The UNION keyword combines the two results into one table.

Table1: First Table in Database.
Table2: Second Table in Database.
column_match: The column common to both the tables.

Non Equijoin

```
SELECT Name, Address
FROM Buyer
LEFT JOIN Property
ON Price < MaxPrice;
```

Compares columns with an operator other than =, such as < and >.

Self Join

```
SELECT A.Name, B.Name
FROM Employ eeM anager A
INNER JOIN Employ eeM anager B
ON B.ID = A.Manager;
```

Joins a table to itself. A self-join can compare any columns of a table, as long as the columns have comparable data types. If a foreign key and the referenced primary key are in the same table, a self-join commonly compares those key columns. In a self-join, aliases are necessary to distinguish left and right tables.

Cross Join

```
SELECT Model, Gigabytes, IPhone.Price +
Storage.Price
FROM IPhone
CROSS JOIN Storage;
```

Combines two tables without comparing columns. Uses a CROSS JOIN clause without an ON clause. As a result, all possible combinations of rows from both tables appear in the result.

SubQuery

```
SELECT Language, Percentage
FROM Countr yLa nguage
WHERE Percentage >
         (SELECT Percentage
        FROM Countr yLa nguage
         WHERE Countr yCode = 'ABW'
                AND IsOfficial = 'T');
```

Sometimes called a **nested query** or **inner query**, is a query within another SQL query. Subquery runs first.

The subquery is typically used in a SELECT statement's WHERE clause to return data to the outer query and restrict the selected results. The subquery is placed inside parentheses ().

Correlated Subquery

```
SELECT Name, CountryCode, Population
FROM City C
WHERE Population >
        (SELECT AVG(Po pul ation)
       FROM City
        WHERE Countr yCode = C.Coun try Code);
```

When the subquery's WHERE clause references a column from the outer query. The rows selected depend on what row is currently being examined by the outer query.

If a column name is identical to a column name in the outer query, the TableName.ColumnName differentiates the columns. Ex: City.C-ountryCode refers to the City table's CountryCode column. An alias can also help differentiate the columns.

Exists Operator

```
SELECT Name, CountryCode
FROM City C
WHERE EXISTS
        (SELECT *
       FROM Countr yLa nguage
        WHERE Countr yCode = C.Coun tryCode
              AND Percentage > 97);
```

Returns TRUE if a subquery selects at least one row and FALSE if no rows are selected. The NOT EXISTS operator returns TRUE if a subquery selects no rows and FALSE if at least one row is selected.

Flatten Subquery

```
SELECT Name
FROM Country
WHERE Code IN
      (SELECT Countr yCode
       FROM City
        WHERE Population > 1000000);
```

Flatten Subquery (cont)

```
-- REPLACED BY
SELECT DISTINCT Name
FROM Country
INNER JOIN City ON Code = Countr yCode
WHERE Population > 1000000;
```

Replacing with a join. Most subqueries that follow NOT EXISTS or contain a GROUP BY clause cannot be flattened.

Steps:
1. Retain the outer query SELECT, FROM, GROUP BY, HAVING, and ORDER BY clauses.
2. Add INNER JOIN clauses for each subquery table.
3. Move comparisons between subquery and outer query columns to ON clauses.
4. Add a WHERE clause with the remaining expressions in the subquery and outer query WHERE clauses.
5. Remove duplicate rows with SELECT DISTINCT.