



Qizmt SQL Reference Guide

Version: Alpha 1.2
Revision: 2010-02-02

Contents

Getting Help on Qizmt SQL Commands	5
Qizmt SQL Commands.....	5
SELECT	5
CREATE TABLE	7
CHAR(N)	7
INT	7
LONG	7
DOUBLE	7
DATETIME.....	8
INSERT	8
INSERT IMPORT	9
INSERT BIND	9
INSERT IMPORTLINES.....	10
ALTER TABLE RENAME SWAP.....	10
DELETE.....	10
TRUNCATE TABLE	10
DROP TABLE	11
SHELL.....	11
Qizmt SQL Aggregate Function	12
AVG	12
BIT_AND	12
BIT_OR.....	12
BIT_XOR.....	13
CHOOSERND.....	13
COUNT.....	13
COUNTDISTINCT.....	13
FIRST.....	13
LAST.....	13
MAX.....	13
MIN.....	13
STD	14
STD_SAMP.....	14
SUM.....	14
VAR_POP	14

VAR_SAMP 14

Qizmt SQL Functions 14

ABS 15

ACOS..... 15

ADD_MONTHS 15

ASIN 15

ATAN 15

ATN2..... 15

CAST 15

CEILING..... 16

CHARINDEX 16

COMPARE..... 16

CONCAT..... 16

COS 16

COT..... 16

DATEADD..... 17

DATEDIFF 17

DEGREES..... 17

EQUAL 17

EXP 18

FLOOR..... 18

FORMAT 18

GREATER..... 18

GREATEREQUAL 18

INSTR 18

ISNOTNULL 18

ISNULL 19

LAST_DAY 19

LEFT 19

LEN 19

LESSER 19

LESSEREQUAL..... 19

LOG..... 19

LOG10..... 20

LOWER..... 20

LPAD 20

LTRIM 20

Qizmt SQL Reference Guide

MOD 20

MONTHS_BETWEEN..... 20

NEXT_DAY 20

NOTEQUAL 21

NULLIF 21

NVL 21

NVL2 21

PATINDEX 21

ISLIKE 21

PI 22

POWER 22

RADIANS..... 22

RAND 22

REPLACE 22

REVERSE 22

RIGHT 23

ROUND 23

RPAD..... 23

RTRIM..... 23

SIGN..... 23

SIN 23

SPACE 23

SQRT 23

SQUARE..... 24

SUBSTRING 24

SYSDATE 24

TAN..... 24

TRUNC..... 24

UPPER..... 24

RINDEX 25

CREATE RINDEX..... 25

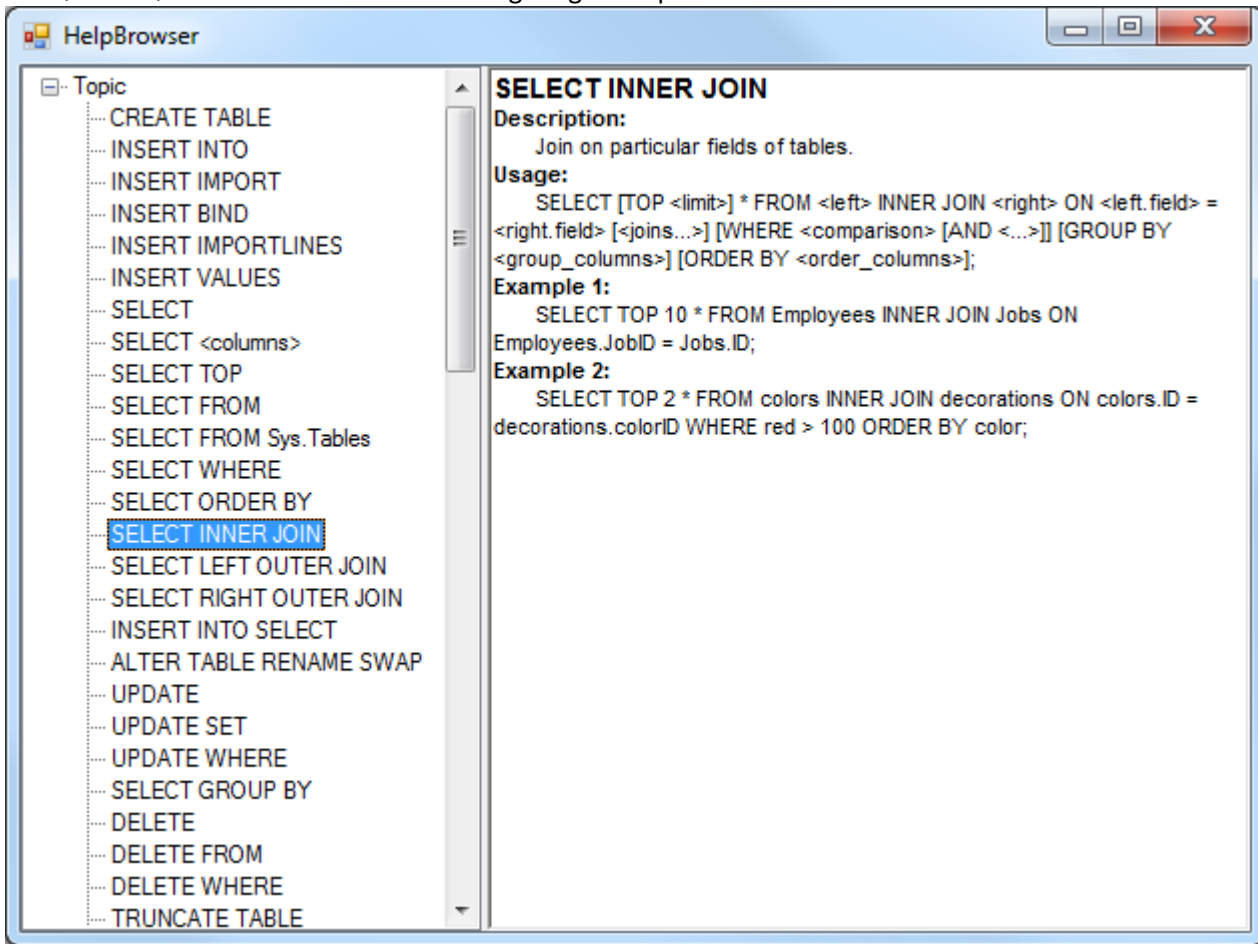
RSELECT 25

BULK UPDATE..... 26

DROP RINDEX 27

Getting Help on Qizmt SQL Commands

In addition to this guide, the SQL help may be obtained by running RDBMS_qa.exe from any machine in the cluster with the Qizmt SQL Extension installed and navigating to help->browse



Qizmt SQL Commands

SELECT

SELECT [TOP <limit>] <[table.]columns/nested scalars/aggregates> FROM [<table_name> [INNER JOIN | LEFT OUTER JOIN | RIGHT OUTER JOIN]...] [WHERE <[table.]comparison/nested scalars/aggregates > [AND <...> | OR <...>]...] [[ORDER BY <[table.]columns>] | [ORDER BY <[table.]columns>]...];

Remarks

SELECT statements are translated into a series of mapreducers on the tables. For example, if performing a complex select on 10TB of data to obtain the top 10 rows, the generated mapreducers will apply the operation on the entire 10TB of data. Indexes, etc. are not considered when executing this statement. It is simple translation to brute force mapreduce.

Scenario	Mapreduce Translation
SIMPLE SELECT	1 mapreduce
SELECT that JOINS 2 tables	2 mapreducers
SELECT that JOINS 5 tables	4 mapreducers

Qizmt SQL Reference Guide

SELECT with ORDER BY or GROUP BY clause	2 mapreducers
SELECT with ORDER BY and GROUP BY clause	3 mapreducers
SELECT TOP that JOINS 5 tables with ORDER BY clause and GROUP BY clause	7 mapreducers
SELECT with lots of nested aggregate and scalar functions	No impact on total number of resulting mapreducers.

It is often better to write custom mapreducers when a SQL query is going to create many of them as the mapreduce solution will likely be more optimal. See the Qizmt Quick Start Guide for a walkthrough on Qizmt mapreduce development.

Example(s)

```
SELECT TOP 10 Name FROM Employees WHERE Years = 4 ORDER BY Name;
```

```
SELECT TOP 2 UPPER(color),red FROM colors WHERE red = 34 ORDER BY color;
```

```
SELECT TOP 1 * FROM Sys.Tables WHERE Table = 'Employees' ORDER BY Table;
```

```
SELECT TOP 10 * FROM Employees INNER JOIN Jobs ON Employees.JobID = Jobs.ID;
```

```
SELECT TOP 2 * FROM colors LEFT OUTER JOIN decorations ON colors.ID = decorations.colorID WHERE red > 100 ORDER BY color;
```

```
SELECT TOP 100
```

```
    FIRST(LargeSQL_Employees.ID),
    CONCAT(CONCAT(SUBSTRING(LAST(FirstName),0,1),'. '),LAST(LastName)),
    FIRST(LargeSQL_Jobs.Name),
    FIRST(YearHired),
    ROUND(DEGREES(ACOS(ASIN(ATN2(ATAN(ASIN(
    COT(COS(SIN(TAN(FIRST(1.1))))),PI()))),2),
    LEN(RPAD(LPAD(REVERSE(LOWER(UPPER(RTRIM(
    LTRIM(REPLACE(SUBSTRING(LAST(FirstName),1,5),'-', ' '))))),10,' '),10,' ')),
    FLOOR(LOG10(POWER(SQRT(EXP(CEILING(
    TRUNC(ABS(LOG(RADIANS(CHOOSERND(5.5))))),1))))),1.1))),
    PATINDEX('e%', LAST(LastName)),
    INSTR('ee', LAST(LastName)),
    AVG(LEN(LargeSQL_Jobs.Name)),
    MIN(LEN(LargeSQL_Jobs.Name)),
    MAX(LEN(LargeSQL_Jobs.Name)),
    BIT_AND(LEN(LargeSQL_Jobs.Name)),
    BIT_OR(LEN(LargeSQL_Jobs.Name)),
    BIT_XOR(LEN(LargeSQL_Jobs.Name)),
    STD(LEN(LargeSQL_Jobs.Name)),
    STD_SAMP(LEN(LargeSQL_Jobs.Name)),
    VAR_POP(LEN(LargeSQL_Jobs.Name)),
    VAR_SAMP(LEN(LargeSQL_Jobs.Name)),
    CHARINDEX('ee', LAST(LastName))
```

```
FROM LargeSQL_Employees
```

```
INNER JOIN LargeSQL_Jobs ON LargeSQL_Employees.JobID = LargeSQL_Jobs.ID
```

```
WHERE
```

```

SUBSTRING(LOWER(LargeSQL_Jobs.Name),1,3) = UPPER('ann')
OR LargeSQL_Jobs.Name LIKE '%teacher%'
OR YearHired >= 2000
OR (YearHired < 1900 AND NOT (NOT MiddleInitial LIKE '%e%'))
GROUP BY LargeSQL_Employees.ID
ORDER BY LastName,FirstName,MiddleInitial,LargeSQL_Employees.ID

```

CREATE TABLE

```
CREATE TABLE <table_name> (<column_name> <type> [, ...]);
```

Remarks

Creates a table with application specific schema of columns. Data types supported include:

CHAR(N)

Description: Data-type for a fixed-length string. String length must be $\leq n$. Representing a single quote in a string literal is done by adding a second adjacent single quote. Storage: $((n * 2) + 1)$ Bytes

Usage: '<text>';

Example 1: 'hello world'

Example 2: 'normal "single quotes" normal'

INT

Description: Data-type for a signed 32-bit integer. An integer literal can include a preceding + or - sign character. Range: -2^{31} (-2,147,483,648) to $2^{31}-1$ (2,147,483,647) Storage: (4 + 1) Bytes

Usage: [+|-]<integer>

Example 1: 9

Example 2: -409983

LONG

Description: Data-type for a signed 64-bit integer, or long integer. A long integer literal can include a preceding + or - sign character. Range: -2^{63} (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807) Storage: (8 + 1) Bytes

Usage: [+|-]<long_integer>

Example 1: 7

Example 2: -2021432403507

DOUBLE

Description: Data-type for a signed 64-bit floating-point number, or double-precision floating-point number. A double literal can include a preceding + or - sign character. Storage: (8 + 1) Bytes

Usage: [+|-]<dec>.<dec>

Example 1: 6.1

Example 2: -300.991

DATETIME

Description: Data-type for date and time. Storage: (8 + 1) Bytes

Usage: '<date time>';

Example 1: '8/7/2010 12:38:15 PM'

Example 2: '8/7/2009 12:38:15 PM'

Example(s)

```
CREATE TABLE Employees (Name char(200), Years int);
```

```
CREATE TABLE colors (color char(50), red int, green int, blue int);
```

INSERT

```
INSERT INTO <table_name> SELECT <columns> FROM <table_name> [WHERE <comparison> [AND <...> | OR <...>]...] [ORDER BY <order_columns>];
```

Remarks

Copy tuples from one or more tables or literal into another table. For inserting literals into another table ad-hoc, see BULK UPDATE. If you have a large amount of data to put into a table, first put it into MR.DFS with then bind it into a table. See fput in qizmt documentation and INSERT BIND commands.

Example(s)

```
INSERT INTO Employees VALUES('George Jefferson', 4);
```

```
INSERT INTO Employees SELECT * FROM NewEmployees ORDER BY Name;
```

```
INSERT INTO EmployeeJobs
SELECT
    FIRST(LargeSQL_Employees.ID),
    CONCAT(CONCAT(SUBSTRING(LAST(FirstName),0,1),'. '),LAST(LastName)),
    FIRST(LargeSQL_Jobs.Name),
    FIRST(YearHired),
    ROUND(DEGREES(ACOS(ASIN(ATN2(ATAN(ASIN(
    COT(COS(SIN(TAN(FIRST(1.1))))),PI()))),2),
    LEN(RPAD(LPAD(REVERSE(LOWER(UPPER(RTRIM(
    LTRIM(REPLACE(SUBSTRING(LAST(FirstName),1,5),'-', ' '))))),10,' '),10,' ')),
    FLOOR(LOG10(POWER(SQRT(EXP(CEILING(
    TRUNC(ABS(LOG(RADIANS(CHOOSERND(5.5))))),1))))),1.1))),
    PATINDEX('e%e', LAST(LastName)),
    INSTR('ee', LAST(LastName)),
    AVG(LEN(LargeSQL_Jobs.Name)),
```

```

MIN(LEN(LargeSQL_Jobs.Name)),
MAX(LEN(LargeSQL_Jobs.Name)),
BIT_AND(LEN(LargeSQL_Jobs.Name)),
BIT_OR(LEN(LargeSQL_Jobs.Name)),
BIT_XOR(LEN(LargeSQL_Jobs.Name)),
STD(LEN(LargeSQL_Jobs.Name)),
STD_SAMP(LEN(LargeSQL_Jobs.Name)),
VAR_POP(LEN(LargeSQL_Jobs.Name)),
VAR_SAMP(LEN(LargeSQL_Jobs.Name)),
CHARINDEX('ee', LAST(LastName))
FROM LargeSQL_Employees
INNER JOIN LargeSQL_Jobs ON LargeSQL_Employees.JobID = LargeSQL_Jobs.ID
WHERE
    SUBSTRING(LOWER(LargeSQL_Jobs.Name),1,3) = UPPER('ann')
    OR LargeSQL_Jobs.Name LIKE '%teacher%'
    OR YearHired >= 2000
    OR (YearHired < 1900 AND NOT (NOT MiddleInitial LIKE '%e%'))
GROUP BY LargeSQL_Employees.ID
ORDER BY LastName,FirstName,MiddleInitial,LargeSQL_Employees.ID

```

INSERT IMPORT

```
INSERT INTO <table_name> IMPORT <dfs_file>;
```

Remarks

Insert data into a table from a MR.DFS rectangular binary data file. The rectangular binary file must have the same schema as the target table for import. This operation translates to a single mapreducer.

Example(s)

```
INSERT INTO Employees IMPORT 'dfs://company-roster';
```

INSERT BIND

```
INSERT INTO <table_name> BIND <dfs_file>;
```

Remarks

Bind data into a table using a MR.DFS rectangular binary data file created using the DbRecordset object. The file is directly bound into the table, leaving the original file name inaccessible. This operation incurs no mapreducers and is used for re-casting MR.DFS data as a SQL table. See the Qizmt SQL Quick Start Guide for more information on DbRecordset.

Example(s)

```
INSERT INTO Employees BIND 'dfs://db-company-roster';
```

INSERT IMPORTLINES

```
INSERT INTO <table_name> IMPORTLINES <dfs_file> [DELIMITER <character>];
```

Remarks

Insert data into a table from a line-based MR.DFS file with a character delimiter. This operation incurs 1 mapreducer.

Example(s)

```
INSERT INTO Employees IMPORTLINES 'dfs://company-roster-lines';
```

ALTER TABLE RENAME SWAP

```
ALTER TABLE <table1> RENAME SWAP <table2>;
```

Remarks

Fast way to swap the names of two tables which have identical schema. If a table is produced by offline sequence, this is a fast way to swap its name with the name of its predecessor which is already serving data.

Example(s)

```
ALTER TABLE Employees RENAME SWAP Employees_alt;
```

DELETE

```
DELETE FROM <table_name> WHERE <comparison> [[AND | OR] <...>];
```

Remarks

Delete rows from a table under certain conditions. See SELECT WHERE for more information on the WHERE clause. This operation performs a full mapreduce on the table but only maps the tuples which match the WHERE clause. For ad-hoc deletes see the BULK UPDATE command.

Example(s)

```
DELETE FROM Employees WHERE ISLIKE '%ferson';
```

TRUNCATE TABLE

```
TRUNCATE TABLE <table_name>;
```

Remarks

Remove all rows from the table. All tuples are deleted in parallel across the cluster. Much faster than DELETE command as no mapreduce overhead is incurred.

Example(s)

```
TRUNCATE TABLE archive_data;
```

DROP TABLE

```
DROP TABLE <table_name>;
```

Remarks

Remove all rows from the table and remove the table from sys.tables. No mapreduce overhead, all tuples are deleted in parallel across the cluster.

Example(s)

```
DROP TABLE archive_data;
```

SHELL

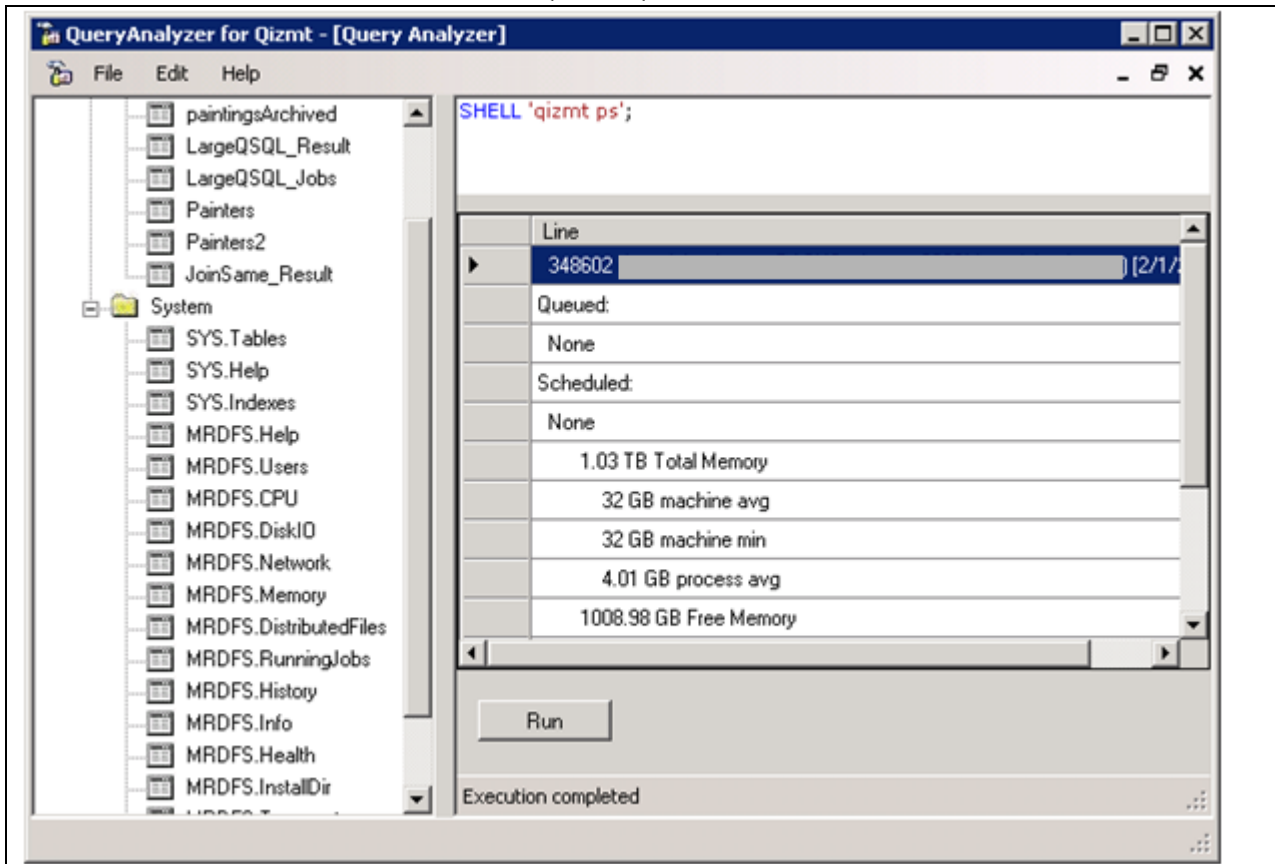
```
SHELL '<program> [<arguments...>]';
```

Remarks

Executes a command on the cluster. Typically, this is used to execute Qizmt commands remotely or as part of a sequence of SQL statements. Newlines from the standard out are persisted and the loss-less standard out may be obtained by concatenating all of the resulting tuples. See the Qizmt documentation for available commands.

Example(s)

```
SHELL 'qizmt ps';
```



Qizmt SQL Aggregate Function

Qizmt SQL Aggregate Function Listings

AVG

Description: Returns the average of the values in a group

Usage: AVG(numeric_expression)

Example 1: SELECT deptID, AVG(year) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, AVG(red) FROM colors GROUP BY blue;

BIT_AND

Description: Performs bitwise AND operations on the values in a group.

Usage: BIT_AND(expression)

Example 1: SELECT angle, BIT_AND(length) FROM Triangles GROUP BY angle;

Example 2: SELECT angle, BIT_AND(base) FROM Trapezoids GROUP BY angle;

BIT_OR

Description: Performs bitwise OR operations on the values in a group.

Usage: BIT_OR(expression)

Example 1: SELECT angle, BIT_OR(length) FROM Triangles GROUP BY angle;

Example 2: SELECT angle, BIT_OR(base) FROM Trapezoids GROUP BY angle;

BIT_XOR

Description: Performs bitwise exclusive OR operations on the values in a group.

Usage: BIT_XOR(expression)

Example 1: SELECT angle, BIT_XOR(length) FROM Triangles GROUP BY angle;

Example 2: SELECT angle, BIT_XOR(base) FROM Trapezoids GROUP BY angle;

CHOOSEMND

Description: Returns a random value from a group.

Usage: CHOOSEMND(expression)

Example 1: SELECT CHOOSEMND(year) FROM Employees GROUP BY deptID;

Example 2: SELECT CHOOSEMND(red) FROM colors GROUP BY blue;

COUNT

Description: Returns the number of items in a group.

Usage: COUNT(expression)

Example 1: SELECT year, COUNT(year) FROM Employees GROUP BY year;

Example 2: SELECT blue, COUNT(blue) FROM colors GROUP BY blue;

COUNTDISTINCT

Description: Returns the number of distinct items in a group.

Usage: COUNTDISTINCT(expression)

Example 1: SELECT year, COUNTDISTINCT(year) FROM Employees GROUP BY year;

Example 2: SELECT blue, COUNTDISTINCT(blue) FROM colors GROUP BY blue;

FIRST

Description: Returns the first value in a group.

Usage: FIRST(expression)

Example 1: SELECT deptID, FIRST(Name) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, FIRST(red) FROM colors GROUP BY blue;

LAST

Description: Returns the last value in a group.

Usage: LAST(expression)

Example 1: SELECT deptID, LAST(Name) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, LAST(red) FROM colors GROUP BY blue;

MAX

Description: Returns the maximum value in a group.

Usage: MAX(expression)

Example 1: SELECT deptID, MAX(year) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, MAX(red) FROM colors GROUP BY blue;

MIN

Description: Returns the minimum value in a group.

Usage: MIN(expression)

Example 1: SELECT deptID, MIN(year) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, MIN(red) FROM colors GROUP BY blue;

STD

Description: Returns the population standard deviation of all values in a group.

Usage: STD(numeric_expression)

Example 1: SELECT deptID, STD(year) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, STD(red) FROM colors GROUP BY blue;

STD_SAMP

Description: Returns the sample standard deviation of all values in a group.

Usage: STD_SAMP(numeric_expression)

Example 1: SELECT deptID, STD_SAMP(year) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, STD_SAMP(red) FROM colors GROUP BY blue;

SUM

Description: Returns the sum of all values in a group.

Usage: SUM(numeric_expression)

Example 1: SELECT angle, SUM(area) FROM Triangles GROUP BY angle;

Example 2: SELECT angle, SUM(area) FROM Trapezoids GROUP BY angle;

VAR_POP

Description: Returns the population variance of all values in a group.

Usage: VAR(numeric_expression)

Example 1: SELECT deptID, VAR_POP(year) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, VAR_POP(red) FROM colors GROUP BY blue;

VAR_SAMP

Description: Returns the sample variance of all values in a group.

Usage: VAR_SAMP(numeric_expression)

Example 1: SELECT deptID, VAR_SAMP(year) FROM Employees GROUP BY deptID;

Example 2: SELECT blue, VAR_SAMP(red) FROM colors GROUP BY blue;

Qizmt SQL Functions

Remarks

Many common SQL scalar and aggregate functions are supported and do not incur any additional mapreduce overhead when used. Functions may be nested in most clauses.

Qizmt SQL Function Listings**ABS**

Description: A mathematical function that returns the absolute (positive) value of the specified numeric expression. Returns the same type as numeric_expression.

Usage: ABS(numeric_expression)

Example 1: SELECT TOP 10 * FROM Employees WHERE ABS(Years) = 4 ORDER BY Name;

Example 2: SELECT TOP 2 name, blue, ABS(red) FROM colors WHERE blue = 34 ORDER BY color;

ACOS

Description: A mathematical function that returns the angle, in radians, whose cosine is the specified numeric expression; also called arccosine. Return type is double.

Usage: ACOS(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE ACOS(angle) = 0.0;

Example 2: SELECT TOP 2 ACOS(angle) FROM Trapezoids WHERE base = 10;

ADD_MONTHS

Description: Returns a datetime with the specified months added.

Usage: ADD_MONTHS(datetime, int)

Example 1: SELECT TOP 100 * FROM Employees WHERE ADD_MONTHS(hireDate, 1) = '1/1/2000';

Example 2: SELECT Name, ADD_MONTHS(hireDate, -6) FROM Employees where EmployeeID = 900;

ASIN

Description: Returns the angle, in radians, whose sine is the specified numeric expression. This is also called arcsine. Return type is double.

Usage: ASIN(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE ASIN(angle) = 0.0;

Example 2: SELECT TOP 2 ASIN(angle) FROM Trapezoids WHERE base = 20;

ATAN

Description: Returns the angle in radians whose tangent is a specified numeric expression. This is also called arctangent. Return type is double.

Usage: ATAN(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE ATAN(angle) = 0.0;

Example 2: SELECT TOP 2 ATAN(angle) FROM Trapezoids WHERE base < 190;

ATN2

Description: Returns the angle, in radians, between the positive x-axis and the ray from the origin to the point (y, x), where x and y are the values of the two specified numeric expressions. Return type is double.

Usage: AT2N(numeric_expression, numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE ATN2(x, y) = 0.0;

Example 2: SELECT TOP 2 ATN2(x, y) FROM Trapezoids WHERE x > 15;

CAST

Description: Converts a value from one type to another.

Usage: CAST(value AS type)

Example 1: SELECT TOP 10 CAST(x AS CHAR(20)) FROM Triangles WHERE ATN2(x, y) = 0.0;

Example 2: SELECT TOP 2 CAST(x AS LONG) FROM Trapezoids WHERE ATN2(x, y) = 0.0;

CEILING

Description: Returns the smallest integral value that is greater than or equal to the specified double number. Return type is double.

Usage: CEILING(double)

Example 1: SELECT TOP 10 * FROM Triangles WHERE CEILING(area) = 10.0;

Example 2: SELECT TOP 2 y, CEILING(area) FROM Trapezoids WHERE y > 90;

CHARINDEX

Description: Searches expression2 for expression1 and returns its starting position if found. The search starts at start_location (0-index based). If expression1 is not found, -1 is returned. Return type is int.

Usage: CHARINDEX(expression1, expression2 [, start_location])

Example 1: SELECT TOP 10 * FROM Employees WHERE CHARINDEX('Mr', title) = 0 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE CHARINDEX('P', color) = 0 ORDER BY color;

COMPARE

Description: Compares expression1 to expression2. Returns 0 if they are the same, less than 0 if expression1 is less than expression2, greater than 0 if expression1 is greater than expression2.

Usage: COMPARE(expression1, expression2)

Example 1: SELECT TOP 100 * FROM Employees WHERE COMPARE(EmployeeID, 900) = 0;

Example 2: SELECT TOP 100 * FROM colors WHERE COMPARE(red, 0) = 0;

CONCAT

Description: Returns a concatenated string.

Usage: CONCAT(char(n), char(m))

Example 1: SELECT CONCAT(Title, CONCAT(', ', Name)) FROM Employees WHERE CHARINDEX('Mr', Title) = 0;

Example 2: SELECT CONCAT(DeptName, CONCAT(', ', Title)) FROM Employees;

COS

Description: A mathematical function that returns the trigonometric cosine of the specified angle, in radians, in specified expression. Return type is double.

Usage: COS(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE COS(angle) = 1.0;

Example 2: SELECT TOP 2 angle, COS(angle) FROM Trapezoids WHERE x = 100;

COT

Description: A mathematical function that returns the trigonometric cotangent of the specified angle, in radians, in the specified expression. Return type is double.

Usage: COT(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE COT(angle) = 1.0;

Example 2: SELECT TOP 2 angle, COT(angle) FROM Trapezoids WHERE base > 89;

DATEADD

Description: Returns a datetime with the specified number interval added to the specified date part.

Usage: DATEADD(char(n) datepart, number int, dt datetime)

Valid datepart:

datepart	Abbreviations
year	yy, yyyy
quarter	q, qq
month	m, mm
day	dd, d
week	wk, ww
hour	hh
minute	mi, n
second	ss, s
millisecond	ms

Example 1: SELECT DATEADD('year', 1, hireDate) FROM Employees WHERE EmployeeID = 900;

Example 2: SELECT TOP 100 * FROM Employees WHERE DATEDIFF('year', DATEADD('month', 6, hireDate), '1/1/2000') = 2;

DATEDIFF

Description: Returns a double that represents the difference in the specified datepart between datetime1 and datetime2.

Usage: DATEDIFF(char(n) datepart, dt1 datetime, dt2 datetime)

Valid datepart:

datepart	Abbreviations
year	yy, yyyy
quarter	q, qq
month	m, mm
day	dd, d
week	wk, ww
hour	hh
minute	mi, n
second	ss, s
millisecond	ms

Example 1: SELECT DATEDIFF('month', hireDate, terminateDate) FROM Employees WHERE active = 0;

Example 2: SELECT TOP 100 * FROM Employees WHERE DATEDIFF('year', DATEADD('month', 6, hireDate), '1/1/2000') = 2;

DEGREES

Description: Returns the corresponding angle in degrees for an angle specified in radians. Return type is double.

Usage: DEGREES(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE DEGREES(angle) = 90.0;

Example 2: SELECT TOP 2 angle, DEGREES(angle) FROM Trapezoids WHERE x > 10;

EQUAL

Description: Compares expression1 with expression2. Returns 1 if they are equal, otherwise, returns 0. Return type is int.

Usage: EQUAL(expression1, expression2)

Example 1: SELECT TOP 10 * FROM Employees WHERE EQUAL(Years, 4) = 1 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE EQUAL(red, blue) = 1 ORDER BY color;

EXP

Description: Returns the exponential value of the specified numeric expression. The constant e is the base of natural logarithms. Return type is double.

Usage: EXP(numeric_expression)

Example 1: SELECT TOP 10 * FROM Employees WHERE EXP(Years) = 1.0 ORDER BY Name;

Example 2: SELECT TOP 2 EXP(red) FROM colors WHERE blue = 9 ORDER BY color;

FLOOR

Description: Returns the largest integer less than or equal to the specified double number. Return type is double.

Usage: FLOOR(double)

Example 1: SELECT TOP 10 * FROM Triangles WHERE FLOOR(area) = 10.0;

Example 2: SELECT TOP 2 * FROM Trapezoids WHERE FLOOR(area) = 1.0;

FORMAT

Description: Returns a string representation of the datetime.

Usage: FORMAT(char(n) format, dt datetime)

Valid Format: This is the same used for C# DateTime format.

Example 1: SELECT TOP 10 FORMAT('dd/MM/yy', hireDate) FROM Employees;

Example 2: SELECT TOP 10 * FROM Employees WHERE FORMAT('yyyy', hireDate) = '1999';

GREATER

Description: Compares expression1 with expression2. Returns 1 if expression1 is greater than expression2, otherwise, returns 0. Return type is int.

Usage: GREATER(expression1, expression2)

Example 1: SELECT TOP 10 * FROM Employees WHERE GREATER(Name, 'M') = 1 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE GREATER(red, blue) = 1 ORDER BY color;

GREATEREQUAL

Description: Compares expression1 with expression2. Returns 1 if expression1 is greater than or equal to expression2, otherwise, returns 0. Return type is int.

Usage: GREATEREQUAL(expression1, expression2)

Example 1: SELECT TOP 10 * FROM Employees WHERE GREATEREQUAL(Name, 'M') = 1 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE GREATEREQUAL(red, blue) = 1 ORDER BY color;

INSTR

Description: Returns the index of the first occurrence of the substring substr in string str.

Usage: INSTR(str, substr)

Example 1: SELECT * FROM Employees WHERE GREATER(INSTR(Title, 'Manager'), -1) = 1;

Example 2: SELECT * FROM colors WHERE GREATER(INSTR(Name, 'purple'), -1) = 1;

ISNOTNULL

Description: Returns 1 if the expression is not null, returns 0 otherwise.

Usage: ISNOTNULL(expression)

Example 1: SELECT TOP 100 * FROM Employees WHERE ISNOTNULL(terminateDate) = 1;

Example 2: SELECT TOP 10 * FROM colors WHERE ISNOTNULL(Name) = 0;

ISNULL

Description: Returns 1 if the expression is null, returns 0 otherwise.

Usage: ISNULL(expression)

Example 1: SELECT TOP 100 * FROM Employees WHERE ISNULL(terminateDate) = 0;

Example 2: SELECT TOP 10 * FROM colors WHERE ISNULL(Name) = 1;

LAST_DAY

Description: Returns the last day of the month.

Usage: LAST_DAY(datetime)

Example 1: SELECT LAST_DAY(payDate) FROM Employees WHERE EmployeeID = 900;

Example 2: SELECT LAST_DAY(hireDate) FROM Employees WHERE EmployeeID = 900;

LEFT

Description: Returns the left part of a character string with the specified number of characters.

Usage: LEFT(char(n), int)

Example 1: SELECT TOP 10 * FROM Employees WHERE LEFT(Name, 4) = 'John' ORDER BY Name;

Example 2: SELECT TOP 2 blue, LEFT(color, 3) FROM colors WHERE blue > 9 ORDER BY color;

LEN

Description: Returns the number of characters of the specified string expression. Return type is int.

Usage: LEN(char(n))

Example 1: SELECT TOP 10 * FROM Employees WHERE LEN(Name) = 4 ORDER BY Name;

Example 2: SELECT TOP 2 LEN(color), red, blue FROM colors WHERE blue > 10 ORDER BY color;

LESSER

Description: Compares expression1 with expression2. Returns 1 if expression1 is less than expression2, otherwise, returns 0. Return type is int.

Usage: LESSER(expression1, expression2)

Example 1: SELECT TOP 10 * FROM Employees WHERE LESSER(Name, 'M') = 1 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE LESSER(red, blue) = 1 ORDER BY color;

LESSEREQUAL

Description: Compares expression1 with expression2. Returns 1 if expression1 is less than or equal to expression2, otherwise, returns 0. Return type is int.

Usage: LESSEREQUAL(expression1, expression2)

Example 1: SELECT TOP 10 * FROM Employees WHERE LESSEREQUAL(Name, 'M') = 1 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE LESSEREQUAL(red, blue) = 1 ORDER BY color;

LOG

Description: Returns the natural logarithm of the specified numeric expression. Return type is double.

Usage: LOG(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE LOG(angle) = 0.0;

Example 2: SELECT TOP 2 LOG(angle), x, y FROM Trapezoids WHERE base = 10;

LOG10

Description: Returns the base-10 logarithm of the specified numeric expression. Return type is double.

Usage: LOG10(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE LOG10(angle) = 0.0;

Example 2: SELECT TOP 2 LOG10(angle), x, y FROM Trapezoids WHERE base = 9;

LOWER

Description: Returns a character expression after converting uppercase character data to lowercase.

Usage: LOWER(char(n))

Example 1: SELECT TOP 10 LOWER(Name) FROM Employees WHERE EmployeeID > 90 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE LOWER(color) = 'red' ORDER BY color;

LPAD

Description: Left pad str to the specified length using padstr. Returns the padded string.

Usage: LPAD(str, length, padstr)

Example 1: SELECT TOP 10 LPAD(Title, 100, ' ') FROM Employees;

Example 2: SELECT TOP 2 * FROM colors WHERE LPAD(Name, 10, ' ') = '.....Blue';

LTRIM

Description: Returns a character expression after it removes leading blanks.

Usage: LTRIM(char(n))

Example 1: SELECT TOP 10 * FROM Employees WHERE LTRIM(Name) = 'john' ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE LTRIM(color) = 'red' ORDER BY color;

MOD

Description: Returns the remainder after dividing numeric_expression1 by numeric_expression2.

Usage: MOD(numeric_expression1, numeric_expression2)

Example 1: SELECT TOP 10 * FROM Triangles WHERE MOD(area, 10) = 0;

Example 2: SELECT TOP 2 MOD(base, 2), area FROM Trapezoids WHERE x = 1;

MONTHS_BETWEEN

Description: Returns a double that represents the number of months between the two datetime operands.

Usage: MONTHS_BETWEEN(datetime1, datetime2)

Example 1: SELECT TOP 100 * FROM Employees WHERE MONTHS_BETWEEN(hireDate, terminateDate) = 6;

Example 2: SELECT TOP 100 BETWEEN(hireDate, terminateDate) FROM Employees ORDER BY EmployeeID;

NEXT_DAY

Description: Returns the next day of the specified datetime.

Usage: NEXT_DAY(datetime)

Example 1: SELECT TOP 10 * NEXT_DAY(payDate) FROM Employees;

Example 2: SELECT TOP 100 * NEXT_DAY(hireDate) FROM Employees;

NOTEQUAL

Description: Compares expression1 with expression2. Returns 1 if they are not equal, otherwise, returns 0. Return type is int.

Usage: NOTEQUAL(expression1, expression2)

Example 1: SELECT TOP 10 * FROM Employees WHERE NOTEQUAL(Years, 4) = 1 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE NOTEQUAL(red, blue) = 1 ORDER BY color;

NULLIF

Description: Returns a null value of the type of expression1 if expression1 is equal to expression2, otherwise returns expression1.

Usage: NULLIF(expression1, expression2)

Example 1: SELECT TOP 10 * NULLIF(hireDate, terminateDate) FROM Employee;

Example 2: SELECT TOP 2 * NULLIF(red, blue) FROM colors;

NVL

Description: Returns expression2 if expression1 is null, otherwise returns expression1.

Usage: NVL(expression1, expression2)

Example 1: SELECT TOP 10 * NVL(MiddleName, ' ') FROM Employees;

Example 2: SELECT TOP 10 * NVL(Address, 'None') FROM Employees;

NVL2

Description: Returns expression3 if expression1 is null, otherwise returns expression2.

Usage: NVL2(expression1, expression2, expression3)

Example 1: SELECT TOP 10 * NVL2(MiddleName, MiddleName, ' ') FROM Employees;

Example 2: SELECT TOP 10 * NVL2(Address, Address, 'None') FROM Employees;

PATINDEX

Description:

Returns the starting position (0-index based) of the first occurrence of a pattern in a specified expression, or -1 if the pattern is not found, on char(n) data types.

A pattern can include regular characters and wildcard characters.

Wildcard character (%): Matches any string of zero or more characters.

Wildcard character (_): Matches any single character.

Wildcard character ([]): Matches any single character within the specified range ([a-f]) or set ([abcdef]).

Wildcard character ([^]): Matches any single character not within the specified range ([^a-f]) or set ([^abcdef]).

Return type is int.

Usage: PATINDEX(char(n) pattern, char(n) text)

Example 1: SELECT TOP 10 * FROM Employees WHERE PATINDEX('%PhD%', Name) = 0 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE PATINDEX('%R%', color) = 0 ORDER BY color;

ISLIKE

Description: Determines whether a specific character string matches a specified pattern. A pattern can include regular characters and wildcard characters. Wildcard character (%): Matches any string of zero or more characters.

Wildcard character(_): Matches any single character. Wildcard character([]): Matches any single character within the specified range ([a-f]) or set ([abcdef]). Wildcard character([^]): Matches any single character not within the specified range ([^a-f]) or set ([^abcdef]). Returns 1 if a match is found, otherwise, returns 0. Return type is int.

Usage: ISLIKE(char(n) text, char(n) pattern)

Example 1: SELECT TOP 10 * FROM Employees WHERE ISLIKE(Name, 'john%') = 1 ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE ISLIKE(color, 'g%') = 1 ORDER BY color;

PI

Description: Returns the constant value of PI. Return type is double.

Usage: PI()

Example 1: SELECT TOP 10 * FROM Triangles WHERE height = PI();

Example 2: SELECT TOP 2 * FROM Trapezoids WHERE height = PI();

POWER

Description: Returns the value of the specified expression to the specified power. Return type is double.

Usage: POWER(numeric_expression, numeric_expression power)

Example 1: SELECT TOP 10 * FROM Triangles WHERE POWER(height, 2) = 4.0;

Example 2: SELECT TOP 2 POWER(height, 3), area FROM Trapezoids WHERE area > 89;

RADIANS

Description: Returns radians when a numeric expression, in degrees, is entered. Return type is double.

Usage: RADIANS(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE RADIANS(angle) = 0.0;

Example 2: SELECT TOP 2 RADIANS(angle), angle FROM Trapezoids;

RAND

Description: Returns a double number greater than or equal to 0.0, and less than 1.0. Return type is double.

Usage: RAND()

Example 1: SELECT TOP 10 * FROM Triangles WHERE area = RAND();

Example 2: SELECT TOP 2 x, y, RAND() FROM Trapezoids WHERE x > y;

REPLACE

Description: Replaces all occurrences of a specified string value with another string value.

Usage: REPLACE(str, pattern, replacement)

Example 1: SELECT TOP 10 REPLACE(Title, 'Trainer', 'T.R.') FROM Employees;

Example 2: SELECT TOP 2 REPLACE(Name, 'v2', '-') FROM colors;

REVERSE

Description: Returns the reverse of a string value.

Usage: REVERSE(char(n))

Example 1: SELECT TOP 10 * FROM Employees WHERE REVERSE(Name) = 'yraM' ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE REVERSE(color) = 'deR' ORDER BY color;

RIGHT

Description: Returns the right part of a character string with the specified number of characters.

Usage: RIGHT(char(n), int)

Example 1: SELECT TOP 10 * FROM Employees WHERE RIGHT(Name, 4) = 'John' ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE RIGHT(color, 3) = 'Red' ORDER BY color;

ROUND

Description: Returns a double, rounded to the specified length or precision in int. Return type is double.

Usage: ROUND(double, int)

Example 1: SELECT TOP 10 * FROM Triangles WHERE ROUND(area, 1) = 10.0;

Example 2: SELECT TOP 2 ROUND(area, 2) FROM Trapezoids WHERE area < 10;

RPAD

Description: Right pad str to the specified length using padstr. Returns the padded string.

Usage: RPAD(str, length, padstr)

Example 1: SELECT TOP 10 RPAD(Title, 100, '- ') FROM Employees;

Example 2: SELECT * FROM colors WHERE RPAD(Name, 10, '. ') = 'Red..... ';

RTRIM

Description: Returns a character string after truncating all trailing blanks.

Usage: RTRIM(char(n))

Example 1: SELECT TOP 10 * FROM Employees WHERE RTRIM(Name) = 'john' ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE RTRIM(color) = 'red' ORDER BY color;

SIGN

Description: Returns the positive (+1), zero (0), or negative (-1) sign of the specified expression. Return type is int.

Usage: SIGN(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE SIGN(angle) = 1;

Example 2: SELECT TOP 2 * FROM Trapezoids WHERE SIGN(angle) = 1;

SIN

Description: Returns the trigonometric sine of the specified angle, in radians, and in numeric expression. Return type is double.

Usage: SIN(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE SIN(angle) = 0.0;

Example 2: SELECT TOP 2 SIN(angle), angle FROM Trapezoids WHERE x = 0.0;

SPACE

Description: Returns a string of repeated spaces.

Usage: SPACE(int)

Example 1: SELECT TOP 10 * FROM Employees WHERE Name = SPACE(1) ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE color = SPACE(1) ORDER BY color;

SQRT

Description: Returns the square root of the specified numeric expression. Return type is double.

Usage: SQRT(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE SQRT(area) = 2.0;

Example 2: SELECT TOP 2 SQRT(area), x, y FROM Trapezoids WHERE x = 3.0;

SQUARE

Description: Returns the square of the specified numeric expression. Return type is double.

Usage: SQUARE(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE SQUARE(area) = 4.0;

Example 2: SELECT TOP 2 SQUARE(area) FROM Trapezoids WHERE base = 9;

SUBSTRING

Description: Returns part of a char(n) expression.

Usage: SUBSTRING(char(n), int startIndex, int length)

Example 1: SELECT TOP 10 * FROM Employees WHERE SUBSTRING(Name, 0, 2) = 'Mr' ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE SUBSTRING(color, 0, 1) = 'R' ORDER BY color;

SYSDATE

Description: Returns the system datetime.

Usage: SYSDATE()

Example 1: SELECT TOP 10 area, SYSDATE() FROM Triangles WHERE SQUARE(area) = 4.0;

Example 2: SELECT TOP 2 angle, SYSDATE() FROM Trapezoids WHERE SQUARE(area) = 9.0;

TAN

Description: Returns the tangent of the input numeric expression. Return type is double.

Usage: TAN(numeric_expression)

Example 1: SELECT TOP 10 * FROM Triangles WHERE TAN(angle) = 0.0;

Example 2: SELECT TOP 2 TAN(angle) FROM Trapezoids WHERE base = 100;

TRUNC

Description: Returns a double, rounded to the specified length or precision in int. Return type is double.

Usage: TRUNC(double, int)

Example 1: SELECT TOP 10 * FROM Triangles WHERE TRUNC(area, 1) = 10.0;

Example 2: SELECT TOP 2 TRUNC(area, 2), base FROM Trapezoids WHERE y = 10.12 AND x < 1;

UPPER

Description: Returns a character expression with lowercase character data converted to uppercase.

Usage: UPPER(char(n))

Example 1: SELECT TOP 10 * FROM Employees WHERE UPPER(Name) = 'JOHN' ORDER BY Name;

Example 2: SELECT TOP 2 * FROM colors WHERE UPPER(color) = 'RED' ORDER BY color;

RINDEX

CREATE RINDEX

```
CREATE RINDEX <indexName> FROM <tableName> [pinMemory | pinMemoryHash | diskonly]
[keepValueOrder] [outlier <delete | random | fifo> <max> ] ON <KeyColumnName>
```

Remarks

Create an RINDEX for the specified table. This command requires the table to already be sorted across the cluster on KeyColumnName. This is usefully when you need to making low latency queries and updates on a very large amount of data without incurring mapreducer overhead.

RINDEX with PINMEMORYHASH are often used for large scale graph processing as they allow very low latency read/write to a large distributed memory source.

Option	When to use
PINMEMORYHASH	When portions of a distributed table are accessed they remain in memory until the RINDEX is dropped. This is enabled by default.
DISKONLY	The table is not cached in memory; heavy disk access every time a query is performed.
KEEPVALUEORDER	Rows returned are in the order they were inserted into the table. By default this order is not preserved.
OUTLIER DELETE	During BULK UPDATE, any tuples of the same key with a quantity exceeding the limit will be deleted.
OUTLIER RANDOM	During BULK UPDATE, any tuples of the same key with a quantity exceeding the limit will be randomly deleted until quantity matches the maximum.
OUTLIER FIFO	During BULK UPDATE, any tuples of the same key with a quantity exceeding the limit will be deleted (older tuples deleted, new tuples kept) until quantity matches the maximum.

Example

```
CREATE RINDEX idxGraph FROM tblGraph ON leftID;
```

RSELECT

```
RSELECT * FROM <indexName> [SAMPLE <sampleSize>] WHERE KEY = <keyValue> [OR KEY = <keyValue>]
```

Remarks

This is a very low latency way to perform ad-hoc queries on very large tables. This can only be done on an RINDEX and the RINDEX can only be created on an already sorted table. (Usually created with a sorted mapreducer job or a SQL with an ORDER BY statement.

In order to use this command, the ADO.NET connection string must specify RINDEX=POOLED.

There is no limit to the number of clients which may RSELECT from an RINDEX in parallel, however RSELECT may not be run in parallel with BULK UPDATE.

The where clause of RSELECT only supports OR statements. The order of tuples returned is random unless KEEPVALUEORDER is specified. If you wish to return a random subset of the tuples, the SAMPLE switch may be used. If a machine is lost during execution, the remaining machines are used to fulfill the query and an exception will be thrown.

Example(s)

```
System.Data.Common.DbProviderFactory fact = DbProviderFactories.GetFactory("Qizmt_DataProvider");

using (DbConnection conn = fact.CreateConnection())
{
    conn.ConnectionString = "Data Source = localhost; RINDEX=POOLED";
    conn.Open();
    DbCommand cmd = conn.CreateCommand();
    cmd.CommandText = "RSELECT * FROM idxGraph WHERE KEY = 1 OR KEY = 99 OR KEY = 434928";
    DbDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        int empID = reader.GetInt32(0);
        string empName = reader.GetString(1);
    }
    reader.Close();
    conn.Close();
}
```

BULK UPDATE

BEGIN BULKUPDATE

[RINSERT INTO indTest VALUES (<literal value>,<literal value>,...) WHERE KEY = <literal value>\0]

|
[RDELETE FROM indTest WHERE KEY = <literal value> AND <column> = <literal value>]

.
.

.

END BULKUPDATE

Remarks

An order of inserts and deletes are used to perform a low latency update. If the table is in memory, both the memory copy and any replicates on disk are updated before execution returns. If a machine is lost during execution, the remaining machines will be used to fulfill the update and an exception will be thrown.

Only one client may bulk-update an RINDEX at a time and no RSELECTs may be run in parallel.

In order to use this command, the ADO.NET connection string must specify RINDEX=POOLED.

Example(s)

```
System.Data.Common.DbProviderFactory fact = DbProviderFactories.GetFactory("Qizmt_DataProvider");

using (DbConnection conn = fact.CreateConnection())
{
    conn.ConnectionString = "Data Source = localhost";
    conn.Open();
    DbCommand cmd = conn.CreateCommand();
```

Qizmt SQL Reference Guide

```
cmd.CommandText = "BEGIN BULKUPDATE\0 " +  
  "RDELETE FROM idxGraph WHERE KEY = 1 \0" +  
  "RINSERT INTO idxGraph VALUES (2, 'Jane') WHERE KEY = 2 \0" +  
  "RDELETE FROM idxGraph WHERE KEY = 3 AND empName = 'Jon' \0" +  
  "END BULKUPDATE";  
cmd.ExecuteNonQuery();  
conn.Close();  
}
```

DROP RINDEX

DROP RINDEX <index name>

Remarks

This command drops the RINDEX on a table but does not delete the table's data. For deleting the tables data see DROP TABLE.

Example(s)

DROP RINDEX idxGraph