SYBASE®
An SAP® Company

Time Series Guide

Sybase IQ 15.4

DOCUMENT ID: DC01154-01-1540-01
LAST REVISED: November 2011

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Audience

This book is intended for RAP – The Trading Edition Enterprise® users who use Sybase® IQ for time series forecasting and analysis. Use this book to get information about time series function SQL syntax and parameters, and information on how the parameters map to the IMSL™ C Stat and C Math third-party external libraries.

For conceptual information on scalar, aggregate, and table user-defined functions, and table parameterized functions, see the *User-Defined Functions Guide*.

Overview

A time series is a sequence of time points, measured at successive times and normally spaced at uniform intervals. Time series forecasting uses a model to forecast future data points based on past data points. Time series analysis identifies the underlying context of the data points, and trends in the data, using techniques such as predictive modeling, autoregressive modeling, and smoothing short-term fluctuations.

Sybase IQ includes SQL functions for time series forecasting and analysis. These functions in turn declare user-defined functions (UDFs) that execute within the IMSL C Stat and C Math external libraries.

The time series forecasting and analysis functions include both OLAP-style aggregates and supporting scalar functions.

Licensing Prerequisites

Time series and forecasting capability is available only with RAP – The Trading Edition Enterprise. All required licenses are included with RAP – The Trading Edition Enterprise.

IMSL Libraries for Time Series Forecasting and Analysis Functions

All time series and forecasting functions except **TS-VWAP** call two third-party external libraries. The IMSL C Stat and C Math libraries, provided by Visual Numerics Inc., contain C functions used for financial time series forecasting and analysis calculations.

The wrapper library **libtsudf** contains user-defined functions (UDFs) that invoke functions contained in the IMSL C Stat and C Math libraries.

The time series and forecasting SQL functions automatically call the libtsudf wrapper library. Sybase IQ loads the IMSL C Stat and C Math libraries when you call a valid user-defined aggregate function for time series and forecasting analysis. Detailed reference information for the IMSL C Stat and C Math libraries is available in *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

The names and locations of the IMSL C Stat and C Math libraries vary, depending on the platform on which Sybase IQ is installed:

Table 1. IMSL Library Locations and File Names

	Windows 64-bit	UNIX (except AIX)	AIX
Directory where libraries are located	bin64	lib64	lib64
Library file name	imslcmath_imsl_dll.dll imslcstat_imsl_dll.dll	libimslcmath_imsl.so libimslcstat_imsl.so	libimslcmath_imsl_r.so libimslcstat_imsl_r.so

Controlling IMSL Library Error Handling and Logging

You can control the error handling and error logging behavior for the time series functions that call the IMSL libraries. If a runtime error occurs when invoking IMSL library functions, Sybase IQ responds according to your error-handling choice, and logs the error according to your error logging choice.

1. Invoke the following **SET OPTION** statement:

   SET OPTION PUBLIC.Time_Series_Error_Level = '*value*'

   Error level *value* is either 0, 1, 2, or 3.
2. Invoke the following **SET OPTION** statement:

   SET OPTION PUBLIC.Time_Series_Log_Level = '*value*'

   Log level *value* is either 0, 1, 2, or 3.

See also

- *Looking Up IMSL Error Codes* on page 6
- *Error Handling Values* on page 4
- *Error Logging Values* on page 5

Error Handling Values

This table lists **Time_Series_Error_Level** values.

Table 2. IMSL Library Error Handling Values

Value	Description
0 (default)	All warnings and errors that can be obtained while invoking an IMSL library function are ignored. When such a condition is encountered, the time series function returns a NULL value.

Value	Description
1	If the time series function obtains a warning or an error message while invoking an IMSL library function, Sybase IQ returns an error message and aborts the SQL query.
2	If the time series function obtains a fatal error message while invoking an IMSL library function, Sybase IQ returns an error message and aborts the SQL query. However, if a warning is obtained, the time series function returns a NULL value.
3	If the time series function obtains a terminal error message while invoking an IMSL library function, Sybase IQ returns an error message and aborts the SQL query. However if a warning or a fatal error is obtained then the time series function returns a NULL value.

See also

- *Error Logging Values* on page 5
- *Controlling IMSL Library Error Handling and Logging* on page 4
- *Looking Up IMSL Error Codes* on page 6

Error Logging Values

This table lists **Time_Series_Log_Level** values.

Table 3. IMSL Library Error Logging Values

Value	Description
0 (default)	All warnings and errors returned while invoking an IMSL library function are ignored and not logged to the log file.
1	If the time series function returns a warning or an error message while invoking an IMSL library function, a message is logged to the log file.
2	If the time series function returns a fatal error message while invoking an IMSL library function, a message is logged to the log file. Warnings are not logged.

Value	Description
3	If the time series function returns a terminal error message while invoking an IMSL library function, a message is logged to the log file. Warnings and fatal errors are not logged.

See also

- *Error Handling Values* on page 4
- *Controlling IMSL Library Error Handling and Logging* on page 4
- *Looking Up IMSL Error Codes* on page 6

Looking Up IMSL Error Codes

IMSL error codes are associated with error messages in the **imslerr.dat** file. Use **imslerr.dat** to obtain an error code description if Sybase IQ returns an IMSL error code.

imslerr.dat is part of the IMSL library and is located in the `bin64` directory of RAPStore. Do not modify the file.

1. Search for `$RAP/RAPStore/<IQ installation directory>/bin64/imslerr.dat`.
2. Open the file and search for the error code. The error code description is provided.

See also

- *Controlling IMSL Library Error Handling and Logging* on page 4
- *Error Handling Values* on page 4
- *Error Logging Values* on page 5

Time Series Forecasting and Analysis Functions

Time series SQL functions are either OLAP-style aggregate functions, or scalar functions that support OLAP-style aggregates. Use the time series and forecasting functions to use autoregressive integrated moving average (ARIMA) or generalized autoregressive conditional heteroskedasticity (GARCH) modeling, and to apply model construction and evaluation utilities, such as a Box-Cox transformation.

Aggregate Time Series Forecasting and Analysis Functions

Aggregate time series forecasting and analysis functions are OLAP-style aggregates designed exclusively for financial time series statistical analysis. As aggregates these functions accept many sets of input values and return a single result.

For information about OLAP and windowing aggregate functions, see *System Administration Guide: Volume 2 > Using OLAP*.

See also

- *TS_ARMA_AR Function [Aggregate]* on page 11
- *TS_ARMA_CONST Function [Aggregate]* on page 13
- *TS_ARMA_MA Function [Aggregate]* on page 15
- *TS_AUTOCORRELATION Function [Aggregate]* on page 17
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_UNI_AR Function [Aggregate]* on page 35
- *TS_BOX_COX_XFORM Function [Aggregate]* on page 37
- *TS_DIFFERENCE Function [Aggregate]* on page 39
- *TS_ESTIMATE_MISSING Function [Aggregate]* on page 44
- *TS_GARCH Function [Aggregate]* on page 46
- *TS_LACK_OF_FIT Function [Aggregate]* on page 52
- *TS_LACK_OF_FIT_P Function [Aggregate]* on page 54
- *TS_MAX_ARMA_AR Function [Aggregate]* on page 56
- *TS_MAX_ARMA_CONST Function [Aggregate]* on page 59
- *TS_MAX_ARMA_LIKELIHOOD Function [Aggregate]* on page 61
- *TS_MAX_ARMA_MA Function [Aggregate]* on page 63
- *TS_OUTLIER_IDENTIFICATION Function [Aggregate]* on page 65
- *TS_VWAP Function [Aggregate]* on page 71

- *TS_PARTIAL_AUTOCORRELATION Function [Aggregate]* on page 69

Aggregate Time Series Function Parameters

This table lists the parameters of each aggregate time series function.

Table 4. Aggregate time series functions

Time series functions	Parameters
TS_ARMA_AR	(*timeseries_expression* , *ar_count*, *ar_elem*, *method*)
TS_ARMA_CONST	(*timeseries_expression* , *method*)
TS_ARMA_MA	(*timeseries_expression* , *ma_count*, *ma_elem*, *method*)
TS_AUTOCORRELATION	(*timeseries_expression* , *lagmax*, *lag_elem*)
TS_AUTO_ARIMA	(*time_value*, *timeseries_expression* [,*max_lag* [,*critical* [,*epsilon* [,*criterion* [,*confidence* [,*model* [,*n_predictions*]]]]]]])
TS_AUTO_ARIMA_OUTLIER	(*time_value*, *timeseries_expression* [,*max_lag* [,*critical* [,*epsilon* [,*criterion* [,*confidence* [,*model* [,*delta*]]]]]]])
TS_AUTO_UNI_AR	(*timeseries_expression* , *ar_count*, *ar_elem*, *method*)
TS_BOX_COX_XFORM	(*timeseries_expression* , *power* [, *shift* [, *inverse*]])
TS_DIFFERENCE	(*timeseries_expression* , *period1* [, *period2* [, *...period 10*]])
TS_ESTIMATE_MISSING	(*timeseries_expression* , *method*)
TS_GARCH	(*timeseries_expression*, *garch_count*, *arch_count*, *xguess_binary_encoding*, [*max_sigma*])
TS_LACK_OF_FIT	(*timeseries_expression* , *p_value*, *q_value*, *lagmax, [tolerance]*)
TS_LACK_OF_FIT_P	(*timeseries_expression* , *p_value*, *q_value*, *lagmax,* [*tolerance]*)
TS_MAX_ARMA_AR	(*timeseries_expression* , *ar_count*, *ar_elem*)
TS_MAX_ARMA_CONST	(*timeseries_expression*)
TS_MAX_ARMA_MA	(*timeseries_expression* , *ma_count*, *ma_elem*)

Time series functions	Parameters
TS_MAX_ARMA_LIKELIHOOD	(*timeseries_expression*)

Scalar Time Series Forecasting and Analysis Functions

Scalar time series and forecasting UDF functions support the **TS_GARCH** and **TS_AUTO_ARIMA** aggregate functions. **TS_GARCH** and **TS_AUTO_ARIMA** each produce a binary composite, but also accept binary inputs. The **TS_INT_ARRAY** function provides inputs for **TS_AUTO_ARIMA** and **TS_AUTO_ARIMA_OUTLIER**; the **TS_DOUBLE_ARRAY** function provides inputs for **TS_GARCH**. The other scalar functions return individual scalar result values from the aggregate functions. The supporting scalar functions map the parameters of the **TS_GARCH** and **TS_AUTO_ARIMA** functions to the parameters of the C functions contained in the external IMSL libraries.

The following scalar functions support the **TS_GARCH** aggregate function.

Table 5. Scalar functions that support TS_GARCH

Time series functions	Parameters
TS_DOUBLE_ARRAY	(*xguess1*, *xguess2*, [… [, *xguess10*] …]])
TS_GARCH_RESULT_A	(*ts_garch_result*)
TS_GARCH_RESULT_AIC	(*ts_garch_result*)
TS_GARCH_RESULT_USER	(*ts_garch_result*, *model_element_number*)

The following scalar functions support the **TS_AUTO_ARIMA** aggregate function:

Table 6. Scalar functions that support TS_AUTO_ARIMA

Time series functions	Parameters
TS_INT_ARRAY	(*int1*, *int2*, *int3*, *int4*, [… [, *int10*] …]])
TS_AUTO_ARIMA_ RESULT_AIC	(*auto_arima_result*)
TS_AUTO_ARIMA_ RESULT_AICC	(*auto_arima_result*)
TS_AUTO_ARIMA_ RESULT_BIC	(*auto_arima_result*)
TS_AUTO_ARIMA_ RESULT_FORECAST_VAL-UE	(*auto_arima_result,model_element_number*)
TS_AUTO_ARIMA_ RESULT_FORECAST_ER-ROR	(*auto_arima_result,forecast_element_number*)
TS_AUTO_ARIMA_ RESULT_MODEL_P	(*auto_arima_result*)

Time series functions	Parameters
TS_AUTO_ARIMA_ RESULT_MODEL_Q	(*auto_arima_result*)
TS_AUTO_ARIMA_ RESULT_MODEL_S	(*auto_arima_result*)
TS_AUTO_ARIMA_ RESULT_MODEL_D	(*auto_arima_result*)
TS_AUTO_ARIMA_RESULT_RESIDUAL_SIG-MA	(*auto_arima_result*)

The following scalar function supports the **TS_AUTO_ARIMA_OUTLIER** aggregate function:

Table 7. Scalar function that supports TS_AUTO_ARIMA_OUTLIER

Time series functions	Parameters
TS_INT_ARRAY	(*int1*, *int2*, *int3*, *int4*, [… [, *int10*] …]])

See also

- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_ VALUE Function [Scalar]* on page 29
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]* on page 31
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]* on page 33
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34
- *TS_DOUBLE_ARRAY Function [Scalar]* on page 43
- *TS_GARCH_RESULT_A Function [Scalar]* on page 48
- *TS_GARCH_RESULT_AIC Function [Scalar]* on page 49
- *TS_GARCH_RESULT_USER [Scalar]* on page 50
- *TS_INT_ARRAY Function [Scalar]* on page 51

Alphabetical List of Functions

This section provides details on all time series functions, including syntax, licensing prerequisites, parameter descriptions, usage, IMSL library mapping, examples, and standards/compatibility information.

TS_ARMA_AR Function [Aggregate]

Calculates the least-square estimates of parameters for an autoregressive moving average (ARMA) model, and returns the requested autoregressive estimate.

Syntax

```
TS_ARMA_AR (timeseries_expression, ar_count, ar_elem, method)
OVER (window-spec)
```

Licensing prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **ar_count –** an integer containing the number of autoregressive values to compute.
- **ar_elem –** an integer identifying the element in the computed AR array that should be returned. *ar_elem* must be greater than 0 and less than or equal to ar_count.
- **method –** (optional) an integer that identifies the type of procedure used to compute estimates. 0 (the default value) = method of least squares; 1 = method of moments.
- **window-spec – TS_ARMA_AR** is an OLAP function requiring an **OVER ()** clause.

Usage

TS_ARMA_AR time series function returns a double-precision floating-point value containing the autoregressive estimate. **TS_ARMA_AR** calls the function **imsls_d_arma** in the IMSL libraries.

IMSL mapping

The arguments of **TS_ARMA_AR** map to the IMSL library function **imsls_d_arma** as follows:

```
params = imsls_d_arma(n_objs, z, p, q, methodID, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **p –** maps to the user-defined aggregate function argument *ar_count*.
- **q –** =1.
- **methodID –** maps to the method argument of **TS_ARMA_AR**. Can be set to either IMSLS_METHOD_OF_MOMENTS or IMSLS_LEAST_SQUARES.

For detailed information on how **imsls_d_arma** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_ARMA_AR** function, and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the first element of an autoregressive estimate consisting of one value from the *data* column using the method of least squares:

```
SELECT TS_ARMA_AR(data,1,1,0) OVER (ORDER BY rownum ROWS BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS res FROM DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 8. Values returned from TS_ARMA_AR

res
0.898793
0.898793
0.898793
0.898793
0.898793
0.898793
0.898793
0.898793
0.898793
...
0.898793

Standards and compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_ARMA_CONST Function [Aggregate]

Calculates the least-square estimates of parameters for an autoregressive moving average (ARMA) model, and returns an estimated constant.

Syntax

```
TS_ARMA_CONST (timeseries_expression, method)
OVER (window-spec)
```

Licensing prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **method –** an integer that identifies the type of procedure used to compute estimates. 0 (the default value) = method of least squares; 1 = method of moments.
- **window-spec – TS_ARMA_CONST** is an OLAP function requiring an **OVER ()** clause.

Usage

This time series function returns a double-precision floating-point value containing the constant estimate produced by the function. **TS_ARMA_CONST** calls the function **imsls_d_arma** in the IMSL libraries.

IMSL mapping

The arguments of **TS_ARMA_CONST** map to the IMSL library function **imsls_d_arma** as follows:

```
params = imsls_d_arma(n_objs, z, p, q, IMSLS_CONSTANT, method_id, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **p –** =1.
- **q –** =1.
- **methodID –** maps to the method argument of **TS_ARMA_CONST**.

For detailed information on how the function **imsls_d_arma** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example 1

This example shows a SQL statement containing the **TS_ARMA_CONST** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns an estimated constant from the *data* column using the method of least squares:

```
SELECT TS_ARMA_CONST(data,0) OVER (ORDER BY ROWNUM rows BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS res FROM DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 9. Values returned from TS_ARMA_CONST example 1

res
0.082077
0.082077
0.082077
0.082077
0.082077
0.082077
0.082077
0.082077
0.082077
...
0.082077

Example 2

This example provides a sample query that returns estimates for the AR, MA, and constant parameters. The first element for AR and MA in the array contains one element.

```
select ts_arma_ar(data,1,1,0) over (order by rownum rows between
unbounded preceding and unbounded following) as ar_param,
ts_arma_ma(data,1,1,0) over (order by rownum rows between unbounded
preceding and unbounded following) as ma_param, ts_arma_const(data,
0) over (order by rownum rows between unbounded preceding and
unbounded following) as const_param FROM DATASET
```

Sybase IQ returns 50 rows of data, each containing the same three values:

Table 10. Values returned from TS_ARMA_CONST example 2

ar_param	ma_param	const_param
0.898793	0.105075	0.082077

ar_param	ma_param	const_param
0.898793	0.105075	0.082077
0.898793	0.105075	0.082077
0.898793	0.105075	0.082077
0.898793	0.105075	0.082077
0.898793	0.105075	0.082077
0.898793	0.105075	0.082077
0.898793	0.105075	0.082077
0.898793	0.105075	0.082077
...	...	...
0.898793	0.105075	0.082077

Standards and compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_ARMA_MA Function [Aggregate]

Calculates the least-square estimates of parameters for an autoregressive moving average (ARMA) model, and returns the requested moving average estimate.

Syntax

```
TS_ARMA_MA (timeseries_expression, ma_count, ma_elem, method)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **ma_count –** an integer containing the number of autoregressive values to compute.

- **ma_elem –** an integer identifying the element to return from the computed moving average array. The integer must be greater than 0 and less than or equal to ma_count.
- **method –** (optional) an integer identifying the procedure to use to calculate estimates. 0 (the default value) = method of least squares and 1 = method of moments.
- **window-spec – TS_ARMA_MA** is an OLAP function requiring an **OVER ()** clause.

Usage

This time series function returns a double-precision floating-point value representing the moving average estimate. **TS_ARMA_MA** calls the function **imsls_d_arma** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_ARMA_MA** map to the IMSL library function **imsls_d_arma** as follows:

```
params = imsls_d_arma(n_objs, z, p, q, method_id, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **p –** =1.
- **q –** maps to the user-defined aggregate function argument ma_count.
- **method_id –** maps to the method argument of **TS_ARMA_MA**.

For detailed information on how the function **imsls_d_arma** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_ARMA_MA** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the first element of an array containing one element from the *data* column using the method of least squares:

```
SELECT TS_ARMA_MA(data,1,1,0) OVER (ORDER BY rownum ROWS BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS res FROM DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 11. Values Returned from TS_ARMA_MA

res
0.105075
0.105075
0.105075
0.105075

res
0.105075
0.105075
0.105075
0.105075
0.105075
...
0.105075

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_AUTOCORRELATION Function [Aggregate]

Calculates the sample autocorrelation function of a stationary time series.

Syntax

```
TS_AUTOCORRELATION (timeseries_expression, lagmax, lag_elem)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **lagmax –** an integer specifying the maximum lag of autocovariances, autocorrelations, and standard errors of autocorrelations. The integer must be greater than or equal to 1, and less than the number of elements in the time series.
- **lag_elem –** an integer specifying which element in the autocorrelation array is to be returned. The integer must be greater than zero, and less than or equal to lagmax.

- **window-spec – TS_AUTOCORRELATION** is an OLAP function requiring an **OVER ()** clause.

Usage

This time series function returns a double-precision floating-point value representing the autocorrelation value. **TS_AUTOCORRELATION** calls the function **imsls_d_autocorrelation** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_AUTOCORRELATION** map to the IMSL library function **imsls_d_autocorrelation()** as follows:

```
params = imsls_d_autocorrelation(n_objs, x[], lagmax, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **x[] –** contains the value of *timeseries_expression* for the current window frame.
- **lagmax –** maps to the user-defined aggregate function argument *lag_max*.

For detailed information on how the function **imsls_d_autocorrelation** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_AUTOCORRELATION** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the second element from an array containing autocorrelations of the time series data from the *data* column:

```
SELECT TS_AUTOCORRELATION(data,2,2) OVER (ORDER BY ROWNUM rows
BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS res FROM
DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 12. Values Returned from TS_AUTOCORRELATION

res
0.803659
0.803659
0.803659
0.803659
0.803659
0.803659

res
0.803659
0.803659
0.803659
...
0.803659

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_AUTO_ARIMA Function [Aggregate]

Determines parameters of a multiplicative seasonal autoregressive integrated moving average (ARIMA) model, and produces forecasts that incorporate the effects of outliers that have effects that persist beyond the end of the series.

Syntax

```
TS_AUTO_ARIMA( <time_value>, <timeseries_expression> [ , <max_lag> [ ,
<critical    > [ , epsilon> [ , <criterion> [ , <confidence> [, <model> [ ,
<n_predictions>]]]]]] )
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **time_value –** the time value for each input time-series data point.
- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series to be differenced.
- **max_lag –** (optional) represents the maximum lag allowed when fitting an AR(p) model. Must be an integer constant or constant expression. The default is 10.

- **critical –** (optional) critical value used as a threshold for outlier detection. The value must be greater than 0, and must be a double-precision floating point constant or constant expression. The default is 3.0.
- **epsilon –** (optional) positive tolerance value controlling the accuracy of parameter estimates during outlier detection. The value must be a double-precision floating point constant or constant expression. The default is 0.001.
- **criterion –** (optional) the information criterion used for optimum model selection. Must be an integer constant or constant expression valued at 0, 1, or 2:
  - 0 – (default) Akaike's Information Criterion (AIC)
  - 1 – Akaike's Corrected Information Criterion (AICC)
  - 2 – Bayesian Information Criterion (BIC)
- **confidence –** (optional) confidence level for computing forecast confidence limits, taken from the exclusive interval (0, 100). Typical choices for confidence are 90.0, 95.0, and 99.0. Must be a double-precision floating point constant or constant expression. The default is 95.0.
- **n_predictions –** (optional) the number of forecasts requested. Must be a positive integer constant or constant expression. The maximum supported value is 2000. The default is 0.
- **model –** (optional) represents a binary encoded integer array, with a length of four integers, containing the ARIMA values for *p*, *q*, *s*, and *d*, in that order. If the *model* argument is specified in the **TS_INT_ARRAY** function, then the **TS_AUTO_ARIMA** function determines the *p*, *q*, *s*, and *d*, values using the method *3* (**Specified ARIMA)** parameter of the (*IMSLS_METHOD, int method*) argument of the *imsls_d_auto_arima* function in the IMSL library. If the *model* parameter is **NULL**, then the **TS_AUTO_ARIMA** function automatically calculates an ARIMA(p,0,0)X(0,d,0) model, which minimizes the specified error criterion. The default is NULL.
- **window-spec – TS_AUTO_ARIMA** is an OLAP function requiring an **OVER()** clause containing an **ORDER BY** clause. **ROWS** or **RANGE** specifiers are not allowed in the **OVER()** clause.

Usage

As an OLAP-style aggregate function, **TS_AUTO_ARIMA** produces a single SQL result—a specially encoded variable-length binary result value. Supporting scalar functions accept the binary composite output value and return individual scalar result values from it.

Since an OLAP-style aggregate function returns one result value per input tuple, the same binary composite result is returned for each row within a partition. If you do not specify a **PARTITION BY** clause in the **OVER** clause, use **SELECT FIRST** to reduce the results down to a single tuple containing the binary composite result. If you do specify a **PARTITION BY** clause in the **OVER** clause, use **SELECT DISTINCT** to eliminate all but one tuple per partition.

IMSL Mapping

Mappings to the parameters of the IMSL C functions in the external VNI library are performed by the **TS_AUTO_ARIMA_RESULT** supporting scalar functions.

Example

In this example, Sybase IQ computes the AUTO_ARIMA model independently for each of the four stock symbols. The **DISTINCT** qualifier reduces the set of tuples to one tuple per stock symbol. Finally, one output row is returned containing the stock symbol and all the relevant information describing the AUTO_ARIMA model computed for that stock.

```
select stock_symbol,
TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA( auto_arima_res ),
    TS_AUTO_ARIMA_RESULT_AIC( auto_arima_res),
    TS_AUTO_ARIMA_RESULT_AICC( auto_arima_res),
    TS_AUTO_ARIMA_RESULT_BIC( auto_arima_res),
TS_AUTO_ARIMA_RESULT_FORECAST_VALUE( auto_arima_res, 1),
    TS_AUTO_ARIMA_RESULT_FORECAST_ERROR( auto_arima_res, 1),
TS_AUTO_ARIMA_RESULT_FORECAST_VALUE( auto_arima_res, 2),
    TS_AUTO_ARIMA_RESULT_FORECAST_ERROR( auto_arima_res, 2),

TS_AUTO_ARIMA_RESULT_FORECAST_VALUE( auto_arima_res, 3),

    TS_AUTO_ARIMA_RESULT_FORECAST_ERROR( auto_arima_res, 3),

TS_AUTO_ARIMA_RESULT_MODEL_P( auto_arima_res),
    TS_AUTO_ARIMA_RESULT_MODEL_Q( auto_arima_res),

    TS_AUTO_ARIMA_RESULT_MODEL_S( auto_arima_res),

    TS_AUTO_ARIMA_RESULT_MODEL_D( auto_arima_res)
from ( select distinct
            stock_symbol,
            TS_AUTO_ARIMA(stock_trade_time,
                trade_price,

                1, 3.0, 4.0, 0, 95.0, TS_INT_ARRAY(4, 0,
                1, 0, 3))
        over (partition by stock_symbol
                order by stock_trade_time)
        as auto_arima_res
from stock_trades
where stock_symbol in ('XYZ', 'XZZ', 'ZXZ', 'ZZZ')
) as auto_arima_per_stock
```

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26

TS_AUTO_ARIMA_OUTLIER Function [Aggregate]

TS_AUTO_ARIMA_OUTLIER accepts an input time series and automatically determines the parameters of a multiplicative seasonal autoregressive integrated moving average (ARIMA) model. Whereas **TS_AUTO_ARIMA** uses the ARIMA model to forecast values beyond the set of inputs,**TS_AUTO_ARIMA_OUTLIER** uses the ARIMA model to identify statistical outliers in the input time series, and returns the outlier type of each one.

Syntax

```
TS_AUTO_ARIMA_OUTLIER( <time_value>, <timeseries_expression> [, <max_lag> [,
<critical    > [, <epsilon> [, <criterion> [, <confidence> [, <model> [,
<delta>]]]]]]] )
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **time_value –** the time value for each input time-series data point.
- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series to be differenced.
- **max_lag –** (optional) represents the maximum lag allowed when fitting an AR(p) model. Must be a positive integer constant or constant expression. The default is 10.
- **critical –** (optional) critical value used as a threshold for outlier detection. The value must be greater than 0, and must be a double-precision floating point constant or constant expression. The default is 3.0.
- **epsilon –** (optional) positive tolerance value controlling the accuracy of parameter estimates during outlier detection. Must be a double-precision floating point constant or constant expression. The default is 0.001.
- **criterion –** (optional) the information criterion used for optimum model selection. Must be an integer constant or constant expression valued at 0, 1, or 2. The default is 0.
  - 0 – (default) Akaike's Information Criterion (AIC)
  - 1 – Akaike's Corrected Information Criterion (AICC)
  - 2 – Bayesian Information Criterion (BIC)

- **confidence –** (optional) confidence level for computing forecast confidence limits, taken from the exclusive interval (0, 100). Typical choices for confidence are 90.0, 95.0, and 99.0. Must be a double-precision floating point constant or constant expression. The default is 95.0.
- **delta –** (optional) the dampening effect parameter used in the detection of a temporary change outlier. Must be a double-precision floating point constant or constant expression. The default is 0.7.
- **model –** (optional) represents a binary-encoded integer array, with a length of four integers, containing the ARIMA values for *p*, *q*, *s*, and *d*, in that order. If the *model* argument is specified in the **TS_INT_ARRAY** function, then the **TS_AUTO_ARIMA_OUTLIER** function determines the *p*, *q*, *s*, and *d*, values using the method *3* (**Specified ARIMA)** parameter of the (*IMSLS_METHOD, int method*) argument of the *imsls_d_auto_arima* function in the IMSL library. If the *model* parameter is **NULL**, then the **TS_AUTO_ARIMA_OUTLIER** function automatically calculates an ARIMA(p, 0,0)X(0,d,0) model, which minimizes the specified error criterion. The default is NULL.
- **window-spec – TS_AUTO_ARIMA_OUTLIER** is an OLAP function requiring an **OVER()** clause containing an **ORDER BY** clause. **ROWS** or **RANGE** specifiers are not allowed in the **OVER()** clause

Usage

The inputs to **TS_AUTO_ARIMA** and **TS_AUTO_ARIMA_OUTLIER** are nearly identical. However, **TS_AUTO_ARIMA_OUTLIER** returns different values for each input row within a partition, while **TS_AUTO_ARIMA** returns the same value for every row. Because of these differences in result data type and result value scoping, you need not use **SELECT FIRST** or **SELECT DISTINCT** to eliminate the duplicate output values. **TS_AUTO_ARIMA_OUTLIER** does not have any supporting scalar functions for decoding results.

Like **TS_AUTO_ARIMA**, **TS_AUTO_ARIMA_OUTLIER** requires the **TS_INT_ARRAY** supporting scalar function. **TS_INT_ARRAY** provides the binary composite input.

The function returns an integer value for each input tuple, specifying the type of outlier for the time and data value within each tuple. If the time and data value is not an outlier, the function returns NULL. The integer values are:

- **0 –** innovational outlier (IO).
- **1 –** additive outlier (AO).
- **2 –** level shift (LS).
- **3 –** temporary change (TC).
- **4 –** unable to identify (UI).

See *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library* for detailed information on the five outlier types.

IMSL Mapping

Maps to the outlier identification logic of *imsls_d_auto_arima*.

Example

This example computes the ARIMA model for the time series of the stock prices for XYZ, and then uses that model to identify which of the trades are statistical outliers. For rows which are not identified as outliers, **TS_AUTO_ARIMA_OUTLIER** returns NULL. For rows which are identified as outliers, **TS_AUTO_ARIMA_OUTLIER** returns an integer value between 0 and 4 representing the specific type of outlier for the current row.

```
select stock_trade_time,
    stock_price,
    stock_trade_shares,
    TS_AUTO_ARIMA_OUTLIER(stock_trade_time,
    stock_price)
        over (order by stock_trade_time) as outlier_type
from stock_trades
where stock_symbol = 'XYZ'
```

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_ VALUE Function [Scalar]* on page 29
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]* on page 31
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]* on page 33
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34

TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]

A supporting function for the **TS_AUTO_ARIMA** function. Retrieves the Akaike's Information Criterion (AIC) output parameter produced by **TS_AUTO_ARIMA**.

Syntax

TS_AUTO_ARIMA_RESULT_AIC(*auto_arima_result***)**

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** A varbinary result generated by **TS_AUTO_ARIMA**.

Usage

Returns a double-precision floating point value representing the Akaike's Information Criterion output parameter.

IMSL Mapping

Maps to the *IMSLS_AIC* argument of *imsls_d_auto_arima*.

Example

See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]* on page 29
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]* on page 31
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]* on page 33
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34

TS_AUTO_ARIMA_RESULT_AICC [Scalar]

A supporting function for **TS_AUTO_ARIMA**. Retrieves the corrected AIC (AICC) output parameter produced by **TS_AUTO_ARIMA**.

Syntax

TS_AUTO_ARIMA_RESULT_AICC(*auto_arima_result***)**

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** A varbinary result generated by **TS_AUTO_ARIMA**.

Usage
Returns a double-precision floating point value for the resulting corrected Akaike's Information Criterion output parameter.

IMSL Mapping
Maps to the *IMSLS_AICC* argument of *imsls_d_auto_arima*.

Example
See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]* on page 29
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]* on page 31
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]* on page 33
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34

TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]

A supporting function for **TS_AUTO_ARIMA**. Retrieves the Bayesian Information Criterion (BIC) output parameter produced by **TS_AUTO_ARIMA**.

Syntax

TS_AUTO_ARIMA_RESULT_BIC(*auto_arima_result***)**

Licensing Prerequisites
Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** A varbinary result generated by **TS_AUTO_ARIMA**.

Usage

Returns a double-precision floating point value for the resulting Bayesian Information Criterion output parameter.

IMSL Mapping

Maps to the *IMSLS_BIC* argument of *imsls_d_auto_arima*.

Example

See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_ VALUE Function [Scalar]* on page 29
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]* on page 31
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]* on page 33
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34

TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]

A supporting function for **TS_AUTO_ARIMA**. Retrieves the forecasted standard error values for the original input series produced by **TS_AUTO_ARIMA**.

Syntax

```
TS_AUTO_ARIMA_RESULT_FORECAST_ERROR(auto_arima_result,
forecast_element_number)
```

Licensing Prerequisites
Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** A varbinary result generated by **TS_AUTO_ARIMA**.
- **forecast_element_number –** An integer constant expression value. The permitted range is 1 to n_predictions.

Usage
Returns a double-precision floating point value for the standard error of the specified forecasted element. The implicit time value for this forecasted error value is the last time value in the input *time_values* passed to **TS_AUTO_ARIMA**, plus the specified *forecast_element_number*.

IMSL Mapping
Maps to the *IMSLS_OUTLIER_FORECAST* argument of *imsls_d_auto_arima*.

Example
See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]* on page 29
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]* on page 31
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]* on page 33
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34

TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]

A supporting function for **TS_AUTO_ARIMA**. Retrieves the forecasted values for the requested outlier free series produced by **TS_AUTO_ARIMA**.

Syntax

```
TS_AUTO_ARIMA_RESULT_FORECAST_VALUE(auto_arima_result,
model_element_number)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** a varbinary result generated by **TS_AUTO_ARIMA**.
- **model_element_number –** an integer constant expression value. The permitted range is 1 to n_predictions.

Usage

Returns a double-precision floating point value representing the specified forecast value for the original input series. The implicit time value for this forecasted value is the last time value in the input *time_values* passed to **TS_AUTO_ARIMA**, plus the specified *forecast_element_number*.

IMSL Mapping

Maps to the *IMSLS_OUTLIER_FORECAST* argument of *imsls_d_auto_arima*

Example

See the example for **TS_AUTO_ARIMA** function.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30

TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]

A supporting function for the **TS_AUTO_ARIMA**. Retrieves the *p* value produced by **TS_AUTO_ARIMA** when computing the ARIMA model description.

Syntax

TS_AUTO_ARIMA_RESULT_MODEL_P(*auto_arima_result***)**

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** a varbinary result generated by **TS_AUTO_ARIMA**.

Usage

Returns the double-precision floating point *p* output parameter generated by **TS_AUTO_ARIMA**.

IMSL Mapping

Maps to the first element of the *IMSLS_MODEL* argument of *imsls_d_auto_arima*.

Example

See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]* on page 29

TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]

A supporting function for the **TS_AUTO_ARIMA**. Retrieves the *q* value produced by **TS_AUTO_ARIMA** when computing the ARIMA model description.

Syntax

TS_AUTO_ARIMA_RESULT_MODEL_Q(*auto_arima_result*)

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** a varbinary result generated by **TS_AUTO_ARIMA**.

Usage

Returns the double-precision floating point *q* output parameter generated by **TS_AUTO_ARIMA**.

IMSL Mapping

Maps to the second element of the *IMSLS_MODEL* argument of *imsls_d_auto_arima*.

Example

See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]* on page 29

- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]* on page 33
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34

TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]

A supporting function for the **TS_AUTO_ARIMA**. Retrieves the *s* value produced by **TS_AUTO_ARIMA** when computing the ARIMA model description.

Syntax

TS_AUTO_ARIMA_RESULT_MODEL_S(*auto_arima_result***)**

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** a varbinary result generated by **TS_AUTO_ARIMA**.

Usage

Returns the double-precision floating point *s* output parameter generated by **TS_AUTO_ARIMA**.

IMSL Mapping

Maps to the third element of the *IMSLS_MODEL* argument of *imsls_d_auto_arima*.

Example

See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]* on page 29

TS_AUTO_ARIMA_RESULT_MODEL_D Function [Scalar]

A supporting function for the **TS_AUTO_ARIMA** function. Retrieves the *d* value produced by **TS_AUTO_ARIMA** when computing the ARIMA model description.

Syntax

TS_AUTO_ARIMA_RESULT_MODEL_D(*auto_arima_result***)**

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** a varbinary result generated by **TS_AUTO_ARIMA**.

Usage

Returns the double-precision floating point *s* output parameter generated by **TS_AUTO_ARIMA**.

IMSL Mapping

Maps to the fourth element of the *IMSLS_MODEL* argument of *imsls_d_auto_arima*.

Example

See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30
- *TS_AUTO_ARIMA_RESULT_MODEL_Q Function [Scalar]* on page 31
- *TS_AUTO_ARIMA_RESULT_MODEL_S Function [Scalar]* on page 32
- *TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]* on page 34

TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA Function [Scalar]

A supporting function for the **TS_AUTO_ARIMA**. Retrieves the residual standard error of the outlier-free data points.

Syntax

TS_AUTO_ARIMA_RESULT_RESIDUAL_SIGMA (*auto_arima_result***)**

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **auto_arima_result –** a varbinary result generated by **TS_AUTO_ARIMA**.

Usage

Returns the residual standard error output parameter.

IMSL Mapping

Maps to the *IMSLS_RESIDUAL_SIGMA* argument of *imsls_d_auto_arima*.

Example

See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_AUTO_ARIMA Function [Aggregate]* on page 19
- *TS_AUTO_ARIMA_OUTLIER Function [Aggregate]* on page 22
- *TS_AUTO_ARIMA_RESULT_AIC Function [Scalar]* on page 24
- *TS_AUTO_ARIMA_RESULT_AICC [Scalar]* on page 25
- *TS_AUTO_ARIMA_RESULT_BIC Function [Scalar]* on page 26
- *TS_AUTO_ARIMA_RESULT_FORECAST_ERROR Function [Scalar]* on page 27
- *TS_AUTO_ARIMA_RESULT_FORECAST_VALUE Function [Scalar]* on page 29
- *TS_AUTO_ARIMA_RESULT_MODEL_P Function [Scalar]* on page 30

TS_AUTO_UNI_AR Function [Aggregate]

Performs automatic selection and fitting of a univariate autoregressive time series model.

Syntax

```
TS_AUTO_UNI_AR (timeseries_expression, ar_count, ar_elem, method)
OVER (window-spec)
```

Licensing prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **ar_count –** an integer containing the number of autoregressive values to compute.
- **ar_elem –** an integer identifying which computed autoregressive value to return. The integer must be greater than zero, and less than or equal to *ar_count*.
- **method –** (optional) an integer identifying which method to use when computing AR coefficients, where: 0 = method of moments 1 = method of least squares (the default value) 2 = maximum likelihood.
- **window-spec – TS_AUTO_UNI_AR** is an OLAP function requiring an **OVER ()** clause.

Usage

This time series function returns a double-precision floating-point number containing the autoregressive estimate. **TS_AUTO_UNI_AR** calls the function **imsls_d_auto_uni_ar** in the IMSL libraries.

IMSL mapping

The arguments of **TS_AUTO_UNI_AR** map to the IMSL library function **imsls_d_auto_uni_ar** as follows:

```
params = imsls_d_auto_uni_ar (n_objs, z[], maxlag, p, method, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **maxlag –** maps to the user-defined aggregate function argument *ar_count*.
- **p –** the output parameter, representing the number of autoregressive parameters in the model with minimum AIC.
- **method –** maps to the user-defined aggregate function argument *method*. If ar_elem is greater than p, and if your IMSL library time series function error-handling value is set to

0, Sybase IQ returns a null. If your IMSL library time series function error-handling value is set to a value other than 0, Sybase IQ displays an error message indicating that ar_elem is greater than p.

For detailed information on how the function **imsls_d_auto_uni_ar** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_AUTO_UNI_AR** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the first element from an array containing two elements from the *data* column:

```
SELECT TS_AUTO_UNI_AR(data,2,1,0) OVER (ORDER BY ROWNUM rows BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS res FROM DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 13. Values returned from TS_AUTO_UNI_AR

res
0.883453
0.883453
0.883453
0.883453
0.883453
0.883453
0.883453
0.883453
0.883453
...
0.883453

Standards and compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_BOX_COX_XFORM Function [Aggregate]

Performs a forward or inverse Box-Cox power transformation.

Syntax

```
TS_BOX_COX_XFORM (timeseries_expression, power [, shift [,
inverse] ]) OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **power –** a double-precision floating-point value representing an exponent parameter in the Box-Cox power transformation.
- **shift –** (optional) a double-precision floating-point value representing a shift parameter. The value must satisfy the relation: min(timeseries)+shift>0. Shift defaults to 0.0.
- **inverse –** (optional) a tinyint value; if set to 1, Sybase IQ performs an inverse transformation. If 0 or null, Sybase IQ performs a forward transformation. The default value is 0.
- **window-spec – TS_BOX_COX_XFORM** is an OLAP function requiring an **OVER ()** clause with an unbounded window. **TS_BOX_COX_XFORM** does not support value-based windows; for example, you cannot use a range specifier in the **OVER ()** clause.

Usage

TS_BOX_COX_XFORM returns the corresponding calculated transformed value for each element in the time series; it calls the function **imsls_d_box_cox_transform** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_BOX_COX_XFORM** map to the IMSL library function **imsls_d_box_cox_transform** as follows:

```
params = imsls_d_box_cox_transform(n_objs, z[], power,
IMSLS_SHIFT, shift [, IMSLS_INVERSE], 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **power –** maps to the user-defined aggregate function argument *power*.

- **shift –** maps to the user-defined aggregate function argument *shift*.
- **IMSLS_INVERSE –** if the user-defined aggregate function argument *inverse* is 1, Sybase IQ calls the Box-Cox transform with IMSLS_INVERSE, otherwise this argument is left out of the function call.

For detailed information on how the function **imsls_d_box_cox_transform** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows an input data table, a SQL statement containing the **TS_BOX_COX_XFORM** function, and the data values returned by the function. This example uses the following table (called *BOX_COX_XFORM_DATASET*) as its input data. The BOX_COX_XFORM_DATASET table contains 13 rows of time series data:

Table 14. Input Data Table BOX_COX_XFORM_DATASET

rownum	data
1	7
2	26
3	6
4	60
5	78.5
6	1
7	29
8	15
9	52
10	74.3
11	11
12	56
13	8

The following SQL statement returns the Box-Cox power transformation from the *data* column:

```
SELECT TS_BOX_COX_XFORM(data,1.0,1.0,0) OVER (ORDER BY rownum ROWS
BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS res FROM
BOX_COX_XFORM_DATASET
```

Sybase IQ returns the following 13 rows:

Table 15. Values Returned from TS_BOX_COX_XFORM

res
8
27
7
61
79.5
2
30
16
53
75.3
12
57
9

Standards and compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7

TS_DIFFERENCE Function [Aggregate]

Differences a seasonal or nonseasonal time series.

Syntax

```
TS_DIFFERENCE (timeseries_expression, period1 [, period2 [, ...period
10] ]) OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series to be differenced.
- **period1 ... period10 –** each period is an integer expression containing the period in which the time series is to be differenced. You must specify at least one period, and you can specify up to 10 periods.
- **window-spec – TS_DIFFERENCE** is an OLAP function requiring an **OVER ()** clause with an unbounded window. This function does not support value-based windows; for example, you cannot use a range specifier in the **OVER ()** clause.

Usage

For each element in the time series, **TS_DIFFERENCE** returns the corresponding calculated differenced value for the time series; it calls the function **imsls_d_difference** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_DIFFERENCE** map to the IMSL library function **imsls_d_difference** as follows:

```
params = imsls_d_difference(n_objs, z[], n_differences,
periods [], 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **n_differences –** maps to the *period* arguments defined in **TS_DIFFERENCE**.
- **period –** an array of the *period* arguments defined in **TS_DIFFERENCE**.

For detailed information on how the function **imsls_d_difference** performs time series calculations, see *IMSL C Numerical Library User's Guide Volume 2 of 2: C Stat Library*.

Example

This example shows a SQL statement containing the **TS_DIFFERENCE** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement differences data from the *data* column:

```
SELECT TS_DIFFERENCE(data,1) OVER (ORDER BY ROWNUM rows BETWEEN
UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS res FROM DATASET
```

Sybase IQ returns 50 rows:

Table 16. Values Returned from TS_DIFFERENCE

res
NULL
0.170336
0.191027
1.29692
0.801743
-0.038988
-0.09424
1.61886
-1.12477
1.02925
-1.20614
-0.814478
-0.157049
-1.18213
0.027546
1.45734
-0.990302
-0.833325
-0.637229
-0.08655
-0.12594
-0.122914
-1.39596
0.919785
-0.449474
0.037273

res
-0.954345
-0.562983
1.98379
0.88304
-0.345265
0.934656
0.069088
-0.249428
0.795766
-1.8145
1.27016
1.39266
-0.141794
0.934752
0.982506
0.330772
-1.34311
1.23124
0.209869
0.791146
-0.259155
0.15124
-0.963484
0.383186

Note: The first row of results is NULL because the IMSL library returned a *not a number* (NaN) value for that row.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_DOUBLE_ARRAY Function [Scalar]

A supporting function for **TS_GARCH**. Constructs a logical array consisting of 3 – 10 constant double-precision floating point values, and returns a single varbinary value.

Syntax

```
TS_DOUBLE_ARRAY(xguess1, xguess2, xguess3, [ … [ , xguess10] …] ])
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **xguess –** a set of constant double-precision floating point values. Depending on the **TS_GARCH** values for *p* and *q*, there will be 3 – 10 values in the resulting varbinary-encoded logical array of values.

Usage

A scalar function that supports **TS_GARCH**. Generates the *xguess_array* for the **TS_GARCH** *xguess_binary_encoding* parameter.

IMSL Mapping

Maps to the *float xguess[]* argument of **imsls_d_garch**.

Example

See the example for **TS_GARCH**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_GARCH Function [Aggregate]* on page 46

- *TS_GARCH_RESULT_A Function [Scalar]* on page 48
- *TS_GARCH_RESULT_AIC Function [Scalar]* on page 49
- *TS_GARCH_RESULT_USER [Scalar]* on page 50

TS_ESTIMATE_MISSING Function [Aggregate]

Estimates the missing values in a time series and returns them as a new time series, interspersed with the original time series.

Syntax

```
TS_ESTIMATE_MISSING (timeseries_expression, method)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series to be differenced. If a null-value is provided, it is assumed to reflect a gap in the time series, the value of which will be computed by the function.
- **method –** (optional) An integer specifying the method to use when determining missing values:
  - 0 (default) – estimates the missing time series observations in a gap by the median of the last four time series values before and the first four values after the gap.
  - 1 – uses a cubic spline interpolation method to estimate missing values. Here, the interpolation is again done over the last four time series values before and the first four values after the gap.
  - 2 – assumes that the time series before the gap can be well described by an AR(1) process.
  - 3 – uses an AR(p) model to estimate missing values by a one-step-ahead forecast.
- **window-spec – TS_ESTIMATE_MISSING** is an OLAP function requiring an **OVER ()** clause with an unbounded window. This function does not support value-based windows; for example, you cannot use a range specifier in the **OVER ()** clause.

Usage

Use **TS_ESTIMATE_MISSING** to estimate any missing equidistant time points using one of the four estimation methods. *TS_ESTIMATE_MISSING* calls the function **imsls_d_estimate_missing** in the IMSL libraries

You cannot use **TS_ESTIMATE_MISSING** if more than two consecutive NULL values exist in your set of timepoints. If the first or last two values in the set of timepoints are NULL, the function returns NULL.

IMSL Mapping

The arguments of *TS_ESTIMATE_MISSING* map to the IMSL library function **imsls_d_estimate_missing** as follows:

```
params = imsls_d_estimate_missing(n_objs, tpoints[], z[], method,
0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **tpoints –** an array of indexes for specifying missing values in a sequence of timepoints.
- **z[] –** the accumulated *timeseries_expression*, obtained during calls to next_value.
- **method –** maps to the *method* argument defined in *TS_ESTIMATE_MISSING*.

For detailed information on how the function **imsls_d_estimate_missing** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows an input data table, a SQL statement containing the *TS_ESTIMATE_MISSING* function, and the data values returned by the function. This example uses the following table (called *EST_MISSING_DATASET*) as its input data. The EST_MISSING_DATASET table contains nine rows of time series data:

Table 17. Input Data Table EST_MISSING_DATASET

rownum	data
1	2.8223
2	-0.5721
3	2.2771
4	NULL
5	1.2648
6	1.0278
7	0.6991
8	-1.7539
9	-2.8875

The following SQL statement estimates the value of the data missing from the fourth row:

```
SELECT ts_estimate_missing(data,0) OVER (order by rownum rows
between unbounded preceding and unbounded following) AS res FROM
EST_MISSING_DATASET
```

Sybase IQ returns the following nine rows, replacing the NULL value with 1.0278:

Table 18. Values Returned from TS_ESTIMATE_MISSING

res
2.8223
-0.5721
2.2771
1.0278
1.2648
1.0278
0.6991
-1.7539
-2.8875

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7

TS_GARCH Function [Aggregate]

Used to analyze and forecast volatility in time series data. **TS_GARCH** computes the estimates of the parameters of a GARCH(p, q) model. GARCH (generalized autoregressive conditional heteroskedasticity) is a generalized model of ARCH; the ARCH computation relates the error variance to the square of a previous period's error.

Syntax

```
TS_GARCH (timeseries, garch_count, arch_count, xguess_binary_encoding[,
<max_sigma>])
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series to be differenced.
- **garch_count –** the constant integer number of the p GARCH parameter for the GARCH(p,q) calculation.
- **arch_count –** the constant integer number of the q ARCH parameter for the for the GARCH(p,q) calculation. GARCH count plus ARCH count cannot exceed 9.
- **xguess_binary_encoding –** represents the seed values for the model determination (also known as the *xguess array*), encoded as a constant binary expression generated by a call to the **TS_DOUBLE_ARRAY** scalar function. The array has a length of $p + q + 1$, and contains the initial values for the x parameter used in **TS_GARCH_RESULT_USER**. The first element (the seed for the sigma squared value) must be a non zero positive double-precision floating point value that is less than the value for **max_sigma**. The remaining p and q seed values must be double-precision floating point values that are greater than or equal to zero. Their sum must be less than 1.0.
- **max_sigma –** (optional) a constant double precision floating point upper bound on the sigma squared value. Must be a positive value. The default is 10.0.
- **window-spec – TS_GARCH** is an OLAP function requiring an **OVER()** clause containing an **ORDER BY** clause. **ROWS** or **RANGE** specifiers are not allowed in the **OVER()** clause.

Usage

As an OLAP-style aggregate function, **TS_GARCH** produces a single SQL result—a specially encoded variable-length binary result value. Supporting scalar functions accept the binary composite output value and return individual scalar result values from it.

Since an OLAP-style aggregate function returns one result value per input tuple, the same binary composite result is returned for each row within a partition. If you do not specify a **PARTITION BY** clause in the **OVER** clause, use **SELECT FIRST** to reduce the results to a single tuple containing the binary composite result. If you do specify a **PARTITION BY** clause in the **OVER** clause, use **SELECT DISTINCT** to eliminate all but one tuple per partition.

IMSL Mapping

Mapping to the parameters of the IMSL C functions in the external VNI library is performed by **TS_GARCH_RESULT** supporting scalar functions.

Example

This example computes a GARCH(1,2) model independently for the stock price for each of four specified companies stock prices. The query then uses the DISTINCT qualifier to reduce the set of tuples to one tuple per stock symbol. Finally, one output row is returned containing the stock symbol and all the relevant information describing the GARCH(1,2) model computed for that stock.

The second and third arguments to the **TS_GARCH** function are the *p* and *q* values specifying the GARCH(p,q) type of model is to be used. These values must be positive integer constants or constant expressions.

The fourth argument is the set of seed values for determining the GARCH(p,q) model encoded as a binary constant expression that must be generated via the supporting scalar function **TS_DOUBLE_ARRAY**.

```
select stock_symbol,
    TS_GARCH_RESULT_A( garch_res ) as log_likelihood,
    TS_GARCH_RESULT_AIC( garch_res ) as akaike_info,
    TS_GARCH_RESULT_USER( garch_res, 1) as sigma_squared,
    TS_GARCH_RESULT_USER( garch_res, 2) as q_1,
    TS_GARCH_RESULT_USER( garch_res, 3) as p_1,
    TS_GARCH_RESULT_USER( garch_res, 4) as p_2
from ( select distinct
         stock_symbol,
         TS_GARCH(stock_price, 1, 2,
               TS_DOUBLE_ARRAY(1.2, 0.3, 0.2, 0.3), 4)
            over (partition by stock_symbol
                  order by stock_trade_time) as garch_res
    from stock_trades
    where stock_symbol in ('XYZ', XZZ', 'ZXZ', 'ZZZ')
    as dt1
```

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *TS_DOUBLE_ARRAY Function [Scalar]* on page 43
- *TS_GARCH_RESULT_A Function [Scalar]* on page 48
- *TS_GARCH_RESULT_AIC Function [Scalar]* on page 49
- *TS_GARCH_RESULT_USER [Scalar]* on page 50

TS_GARCH_RESULT_A Function [Scalar]

A supporting function for **TS_GARCH**. Retrieves the log-likelihood output parameter, *A*, produced by the **TS_GARCH** aggregate function.

Syntax

TS_GARCH_RESULT_A (*ts_garch_result* **)**

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **ts_garch_result –** a varbinary result argument generated by a call to the **TS_GARCH** aggregate function.

Usage
Returns a double-precision floating point value for the resulting Log-Likelihood output parameter.

IMSL Mapping
Maps to the *IMSLS_A, float *a,* argument of **imsls_d_garch**.

Example
See the example for **TS_GARCH**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_GARCH Function [Aggregate]* on page 46
- *TS_DOUBLE_ARRAY Function [Scalar]* on page 43
- *TS_GARCH_RESULT_AIC Function [Scalar]* on page 49
- *TS_GARCH_RESULT_USER [Scalar]* on page 50

TS_GARCH_RESULT_AIC Function [Scalar]

A supporting function for **TS_GARCH**. Retrieves the Akaike Information Criterion output parameter, *AIC*, produced by the **TS_GARCH** aggregate function.

Syntax

```
TS_GARCH_RESULT_AIC (ts_garch_result)
```

Licensing prerequisites
Available only with RAP – The Trading Edition Enterprise.

Parameters

- **ts_garch_result –** a varbinary result argument generated by a call to the **TS_GARCH** aggregate function.

Usage
Returns a double-precision floating point value for the resulting Akaike Information Criterion output parameter.

IMSL mapping
Maps to the *IMSLS_AIC, float *aic,* argument of **imsls_d_garch**.

Example
See the example for **TS_GARCH**.

Standards and compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_GARCH Function [Aggregate]* on page 46
- *TS_DOUBLE_ARRAY Function [Scalar]* on page 43
- *TS_GARCH_RESULT_A Function [Scalar]* on page 48
- *TS_GARCH_RESULT_USER [Scalar]* on page 50

TS_GARCH_RESULT_USER [Scalar]

A supporting function for **TS_GARCH**. Accesses each element in the logical array that describes the GARCH(p,q) model.

Syntax

```
TS_GARCH_RESULT_USER (ts_garch_result, model_element_number)
```

Licensing Prerequisites
Available only with RAP – The Trading Edition Enterprise.

Parameters

- **ts_garch_result –** a varbinary result argument generated by a call to the **TS_GARCH** aggregate function.
- **model_element_number –** an integer constant expression value within the range of 1 to (1+*p*+*q*).

Usage
Returns a double-precision floating point value for the specified model description output value. The number of elements returned in the output set is $p + q + 1$. The first element in the

output set is the resulting sigma squared value. The next *q* values are the calculated ARCH parameters. The final *p* values are the determined GARCH parameters.

Although the size of the output set is variable—since the *p* and *q* values must be passed as constant input arguments to **TS_GARCH**—the number of model description values is fixed for any specific **TS_GARCH** call.

IMSL Mapping

Maps to the *IMSLS_RETURN_USER, float x[],* argument of **imsls_d_garch**.

Example

See the example for **TS_GARCH**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9
- *TS_GARCH Function [Aggregate]* on page 46
- *TS_DOUBLE_ARRAY Function [Scalar]* on page 43
- *TS_GARCH_RESULT_A Function [Scalar]* on page 48
- *TS_GARCH_RESULT_AIC Function [Scalar]* on page 49

TS_INT_ARRAY Function [Scalar]

A supporting function for **TS_AUTO_ARIMA** and **TS_AUTO_ARIMA_OUTLIER**.

Syntax

```
TS_INT_ARRAY(int1, int2, int3, int4, [ ... [ , int10] ...] ])
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **int1 ... int10 –** a set of constant integer values. If the *model* parameter of **TS_AUTO_ARIMA** is specified, then supply four integers. Ten integers is the maximum value for the set.

  Note: Integer values are required. Passing noninteger values to the function may cause unexpected results.

Usage

Returns a varbinary value that encodes the specified integer input elements as a logical array of values.

IMSL Mapping
If you supply four integers and pass the result to **TS_AUTO_ARIMA** or **TS_AUTO_ARIMA_OUTLIER**, then **TS_INT_ARRAY** maps to method *3* of the (*IMSLS_METHOD, int method*) input argument of *imsls_d_auto_arima* in the IMSL libraries.

Example
See the example for **TS_AUTO_ARIMA**.

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Scalar Time Series Forecasting and Analysis Functions* on page 9

TS_LACK_OF_FIT Function [Aggregate]

Performs the lack-of-fit test for a univariate time series or transfer function, given the appropriate correlation function.

Syntax

```
TS_LACK_OF_FIT (timeseries_expression, p_value, q_value, lagmax,
[tolerance])
```

```
OVER (window-spec)
```

Licensing Prerequisites
Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **p_value –** an integer containing the number of autoregressive parameters.
- **q_value –** an integer containing the number of moving average parameters.
- **lagmax –** an integer containing the maximum lag of the correlation function.
- **tolerance –** (optional) A floating-point value level used to determine convergence of the nonlinear least-squares algorithm. The default value is 0.
- **window-spec – TS_LACK_OF_FIT** is an OLAP function requiring an **OVER ()** clause.

Usage

This function returns a double-precision floating-point value containing the lack-of-fit statistic (q) for the time series. **TS_LACK_OF_FIT** calls the function **imsls_d_lack_of_fit** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_LACK_OF_FIT** map to the IMSL library function **imsls_d_lack_of_fit** as follows:

```
params = imsls_d_arma(n_objs, z[], p, q,    IMSLS_LEAST_SQUARES,
IMSLS_CONVERGENCE_TOLERANCE, tolerance,   IMSL_RESIDUAL, &residual,
0);correlations = imsls_d_autocorrelation(n_objs-p+lagmax,
residuals, lagmax, 0);result = imsls_d_lack_of_fit(n_objs,
correlations, lagmax, npfree, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **p –** maps to the *p_value* argument defined in **TS_LACK_OF_FIT**.
- **q –** maps to the *q_value* argument defined in **TS_LACK_OF_FIT**.
- **lagmax –** maps to the *lagmax* argument defined in **TS_LACK_OF_FIT**.
- **npfree –** derived from p + q.
- **tolerance –** an optional argument using **IMSLS_CONVERGENCE_TOLERANCE**. If null, the IMSL library applies a default value and does not use **IMSLS_CONVERGENCE_TOLERANCE**.

  For detailed information on how the IMSL function **imsls_d_lack_of_fit** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_LACK_OF_FIT** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the lack of fit statistic on data from the *data* column:

```
select ts_lack_of_fit(data,1,1,5,0.225) over (order by rownum rows
between unbounded preceding and unbounded following) as res from
DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 19. Values Returned from TS_LACK_OF_FIT

res
3.96751

res
3.96751
3.96751
3.96751
3.96751
3.96751
3.96751
3.96751
3.96751
3.96751
...
3.96751

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_LACK_OF_FIT_P Function [Aggregate]

Performs the lack-of-fit test for a univariate time series. This function is identical to **TS_LACK_OF_FIT**, except that **TS_LACK_OF_FIT_P** returns the p-value of q, rather than returning q.

Syntax

```
TS_LACK_OF_FIT_P (timeseries_expression, p_value, q_value, lagmax,
[tolerance])
```

```
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **p_value –** an integer containing the number of autoregressive parameters.
- **q_value –** an integer containing the number of moving average parameters.
- **lagmax –** an integer containing the maximum log of the correlation function.
- **tolerance –** (optional) A floating-point value level used to determine convergence of the nonlinear least-squares algorithm. The default value is 0.
- **window-spec – TS_LACK_OF_FIT_P** is an OLAP function requiring an **OVER ()** clause.

Usage

This function returns a double-precision floating-point value containing the p-value of the lack-of-fit statistic (q) for the time series. **TS_LACK_OF_FIT_P** calls the function **imsls_d_lack_of_fit** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_LACK_OF_FIT_P** map to the IMSL library function **imsls_d_lack_of_fit** as follows:

```
params = imsls_d_arma(n_objs, z[], p, q,    IMSLS_LEAST_SQUARES,
IMSLS_CONVERGENCE_TOLERANCE, tolerance,   IMSL_RESIDUAL, &residual,
0);correlations = imsls_d_autocorrelation(n_objs-p+lagmax,
residuals, lagmax, 0);result = imsls_d_lack_of_fit(n_objs,
correlations, lagmax, npfree, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of *timeseries_expression* for the current window frame.
- **p –** maps to the *p_value* argument defined in **TS_LACK_OF_FIT_P**.
- **q –** maps to the *q_value* argument defined in **TS_LACK_OF_FIT_P**.
- **lagmax –** maps to the *lagmax* argument defined in **TS_LACK_OF_FIT_P**.
- **npfree –** derived from p + q.
- **tolerance –** an optional argument using **IMSLS_CONVERGENCE_TOLERANCE**. If null, the IMSL library applies a default value and does not use **IMSLS_CONVERGENCE_TOLERANCE**.

For detailed information on how the IMSL function **imsls_d_lack_of_fit** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_LACK_OF_FIT_P** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the p value of the lack of fit statistic on data from the *data* column:

```
select ts_lack_of_fit_p(data,1,1,5,0.225) over (order by rownum rows
between unbounded preceding and unbounded following) as res FROM
DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 20. Values Returned from TS_LACK_OF_FIT_P

res
0.735006
0.735006
0.735006
0.735006
0.735006
0.735006
0.735006
0.735006
0.735006
0.735006
...
0.735006

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_MAX_ARMA_AR Function [Aggregate]

Calculates the exact maximum likelihood estimation of the parameters in a univariate ARMA (autoregressive moving average) time series model, and returns the requested autoregressive estimate.

Syntax

```
TS_MAX_ARMA_AR (timeseries_expression, ar_count, ar_elem)
```

```
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **ar_count –** an integer containing the number of autoregressive values to compute.
- **ar_elem –** an integer identifying which element in the computed autoregressive array is to be returned. The integer must be greater than 0 and less than or equal to *ar_count*.
- **window-spec – TS_MAX_ARMA_AR** is an OLAP function requiring an **OVER ()** clause.

Usage

This function returns a double-precision floating-point value containing the autoregressive estimate. *TS_MAX_ARMA_AR* calls the function **imsls_d_max_arma** in the IMSL libraries.

IMSL Mapping

The arguments of *TS_MAX_ARMA_AR* map to the IMSL library function **imsls_d_max_arma** as follows:

```
params = imsls_d_max_arma(n_objs, z[], p, q, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of timeseries_expression for the current window frame.
- **p –** maps to the *ar_count* argument.
- **q –** =1.

For detailed information on how the IMSL function **imsls_d_max_arma** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example 1

This example shows a SQL statement containing the **TS_MAX_ARMA_AR** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the second element from an array containing two autoregressive estimates of data from the *data* column:

```
select ts_max_arma_ar(data,2,2) over (order by rownum rows between
unbounded preceding and unbounded following) as res FROM DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 21. Values Returned from TS_MAX_ARMA_AR Example 1

res
0.179748
0.179748
0.179748
0.179748
0.179748
0.179748
0.179748
0.179748
0.179748
0.179748
...
0.179748

Example 2

This example provides a sample query that returns two columns of results from the DATASET table—the first and second elements of the autoregressive estimates.

```
select ts_max_arma_ar(data,2,1) over (order by rownum rows between
unbounded preceding and unbounded following) as ar_elem1,
ts_max_arma_ar(data,2,2) over (order by rownum rows between
unbounded preceding and unbounded following) as ar_elem2 FROM DATASET
```

Sybase IQ returns 50 rows of data, each containing the same two values:

Table 22. Values Returned from TS_MAX_ARMA_AR Example 2

ar_elem1	ar_elem2
0.731164	0.179748
0.731164	0.179748
0.731164	0.179748
0.731164	0.179748
0.731164	0.179748
0.731164	0.179748

ar_elem1	ar_elem2
0.731164	0.179748
0.731164	0.179748
0.731164	0.179748
0.731164	0.179748
0.731164	0.179748
...	...
0.731164	0.179748

Standards and compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server EnterpriseRAP – The Trading Edition Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_MAX_ARMA_CONST Function [Aggregate]

Calculates the exact maximum likelihood estimation of the parameters in a univariate ARMA (autoregressive moving average) time series model, and returns the constant estimate.

Syntax

```
TS_MAX_ARMA_CONST (timeseries_expression)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **window-spec – TS_MAX_ARMA_CONST** is an OLAP function requiring an **OVER ()** clause.

Usage

This function returns a double-precision floating-point value containing the constant estimate. **TS_MAX_ARMA_CONST** calls the function **imsls_d_arma** in the IMSL libraries.

IMSL Mapping

The arguments of *TS_MAX_ARMA_CONST* map to the IMSL library function **imsls_d_arma** as follows:

```
params = imsls_d_max_arma(n_objs, z, p, q, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of timeseries_expression for the current window frame.
- **p –** = 1.
- **q –** = 1.

For detailed information on how the IMSL function **imsls_d_arma** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_MAX_ARMA_CONST** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the constant estimate of the maximum likelihood autoregressive calculation on data from the *data* column:

```
select ts_max_arma_const(data) over (order by rownum rows between
unbounded preceding and unbounded following) as res FROM DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 23. Values Returned from TS_MAX_ARMA_CONST

res
0.107555
0.107555
0.107555
0.107555
0.107555
0.107555
0.107555
0.107555

res
0.107555
0.107555
...
0.107555

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_MAX_ARMA_LIKELIHOOD Function [Aggregate]

Calculates the exact maximum likelihood estimation of the parameters in a univariate ARMA (autoregressive moving average) time series model, and returns likelihood value (ln) for the fitted model.

Syntax

```
TS_MAX_ARMA_LIKELIHOOD (timeseries_expression)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **window-spec – TS_MAX_ARMA_LIKELIHOOD** is an OLAP function requiring an **OVER ()** clause.

Usage

This function returns a double-precision floating-point value containing the value of -2*(ln(likelihood)). **TS_MAX_ARMA_LIKELIHOOD** calls the function **imsls_d_max_arma** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_MAX_ARMA_LIKELIHOOD** map to the IMSL library function **imsls_d_max_arma** as follows:

```
params = imsls_d_max_arma(n_objs, z, p, q, IMSLS_LOG_LIKELIHOOD,
&likelihood, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of timeseries_expression for the current window frame
- **p –** = 1.
- **q –** = 1.
- **likelihood –** provided by the function call. Contains the log likelihood result.

For detailed information on how the IMSL function **imsls_d_max_arma** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_MAX_ARMA_LIKELIHOOD** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the likelihood value of the maximum likelihood estimation on data from the *data* column:

```
Select ts_max_arma_likelihood(data) over (order by rownum rows
between unbounded preceding and unbounded following) as res FROM
DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 24. Values Returned from TS_MAX_ARMA_LIKELIHOOD

res
-11.7818
-11.7818
-11.7818
-11.7818
-11.7818
-11.7818
-11.7818
-11.7818

res
-11.7818
-11.7818
...
-11.7818

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_MAX_ARMA_MA Function [Aggregate]

Calculates the exact maximum likelihood estimation of the parameters in a univariate ARMA (autoregressive moving average) time series model, and returns the requested moving average estimate.

Syntax

```
TS_MAX_ARMA_MA (timeseries_expression, ma_count, ma_elem)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **ma_count –** an integer containing the number of auto-regressive values to compute.
- **ma_elem –** an integer specifying element in the computed moving average array to return. The integer must be greater than zero, and less than or equal to *ma_count*.
- **window-spec – TS_MAX_ARMA_MA** is an OLAP function requiring an **OVER ()** clause.

Usage

This function returns double-precision floating-point value containing the autoregressive estimate. **TS_MAX_ARMA_MA** calls the function **imsls_d_max_arma** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_MAX_ARMA_MA** map to the IMSL library function **imsls_d_max_arma** as follows:

```
params = imsls_d_max_arma(n_objs, z[], p, q, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of timeseries_expression for the current window frame.
- **p –** =1.
- **q –** maps to the **TS_MAX_ARMA_MA** argument *ma_count*.

For detailed information on how the IMSL function **imsls_d_max_arma** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_MAX_ARMA_MA** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the moving average of the maximum likelihood estimation on data from the *data* column:

```
select ts_max_arma_ma(data,5,4) over (order by rownum rows between
unbounded preceding and unbounded following) as res FROM DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 25. Values Returned from TS_MAX_ARMA_MA

res
-0.035006
-0.035006
-0.035006
-0.035006
-0.035006
-0.035006
-0.035006
-0.035006
-0.035006
-0.035006

res
...
-0.035006

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_OUTLIER_IDENTIFICATION Function [Aggregate]

Detects and determines outliers and simultaneously estimates the model parameters in a time series where the underlying outlier-free series follows a general seasonal or non-seasonal ARMA model.

Syntax

```
TS_OUTLIER_IDENTIFICATION (timeseries_expression, p_value, q_value,
s_value, d_value, [, delta_value[, critical_value]])
```

```
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **p_value –** an integer containing the p-portion of the autoregressive integrated moving average (ARIMA) (p, 0, q)x(0, d, 0)s model that the outlier free series follows.
- **q_value –** an integer containing the q-portion of the ARIMA (p, 0, q)x(0, d, 0)s model that the outlier free series follows.
- **s_value –** an integer containing the s-portion of the ARIMA (p, 0, q)x(0, d, 0)s model that the outlier free series follows.
- **d_value –** an integer containing the d-portion of the ARIMA (p, 0, q)x(0, d, 0)s model that the outlier free series follows.

- **delta_value –** (optional) a double precision float value containing the dampening effect parameter used in detecting a temporary change outlier. The integer must be greater than 0 and less than 1. The default value is 0.7.
- **critical_value –** (optional) a double-precision float value used as a threshold for outlier detection. The default is 3.0.
- **window-spec – TS_OUTLIER_IDENTIFICATION** is an OLAP function requiring an **OVER ()** clause with an unbounded window. This function does not support value-based windows; for example, you cannot use a range specifier in the **OVER ()** clause.

Usage

This function returns an outlier-free time series. **TS_OUTLIER_IDENTIFICATION** calls the function **imsls_d_ts_outlier_identification** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_OUTLIER_IDENTIFICATION** map to the IMSL library function **imsls_d_ts_outlier_identification** as follows:

```
params = imsls_d_ts_outlier_identification(n_objs, model[], z[], 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **model –** an array containing the **TS_OUTLIER_IDENTIFICATION** arguments *p_value*, *s_value*, *q_value*, *d_value*: model[0] = p_value; model[1] = s_value; model[2] = q_value; model[3] = d_value;
- **z[] –** contains the value of timeseries_expression for the current window frame.

If *delta_value* is non-null, the arguments of **TS_OUTLIER_IDENTIFICATION** map to the IMSL library function **imsls_d_ts_outlier_identification** as follows:

```
params = imsls_d_ts_outlier_identification(n_objs, model[], z[],
IMSL_DELTA, delta_value, 0);
```

If *critical_value* is non-null, the arguments of **TS_OUTLIER_IDENTIFICATION** map to the IMSL library function **imsls_d_ts_outlier_identification** as follows:

```
params = imsls_d_ts_outlier_identification(n_objs, model[],
z[],        IMSL_CRITICAL, critical_value, 0);
```

If both *delta_value* and *critical_value* are non-null, the arguments of **TS_OUTLIER_IDENTIFICATION** map to the IMSL library function **imsls_d_ts_outlier_identification** as follows:

```
params = imsls_d_ts_outlier_identification(n_objs, model[],
z[],        IMSL_DELTA, delta_value, IMSL_CRITICAL, critical_value,
0);
```

For detailed information on how the IMSL function **imsls_d_ts_outlier_identification** performs time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows a SQL statement containing the **TS_OUTLIER_IDENTIFICATION** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement detects and determines outliers on data from the *data* column:

```
select ts_outlier_identification(data,1,1,1,1,0.7,3.0) over (order
by rownum rows between unbounded preceding and unbounded following)
as res FROM DATASET
```

Sybase IQ returns 50 rows:

Table 26. Values Returned from TS_OUTLIER_IDENTIFICATION

res
0.315523
0.485859
0.676886
1.97381
2.77555
2.73657
2.64233
4.26118
3.13641
4.16566
2.95952
2.14504
1.98799
0.805859
0.833405
2.29075
1.30045
0.467122

res
-0.170107
-0.256657
-0.382597
-0.505511
-1.90147
-0.981688
-1.43116
-1.39389
-2.34823
-2.91122
-0.927423
-0.044383
-0.389648
0.545008
0.614096
0.364668
1.16043
-0.654063
0.616094
2.00875
1.86696
2.80171
3.78422
4.11499
2.77188
4.00312
4.21298

res
5.00413
4.74498
4.89621
3.93273
4.31592

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_PARTIAL_AUTOCORRELATION Function [Aggregate]

Calculates the sample partial autocorrelation function of a stationary time series.

Syntax

```
TS_PARTIAL_AUTOCORRELATION (timeseries_expression, lagmax, lag_elem)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **timeseries_expression –** a numeric expression, generally a column name, containing an element in a time series.
- **lagmax –** an integer containing the maximum lag of autocovariance, autocorrelations, and standard errors of autocorrelations to be calculated. The integer must be greater than or equal to 1, and less than the number of elements in the time series.
- **lag_elem –** an integer identifying the element in the autocorrelation array to return. The integer must be greater than 0 and less than or equal to *lagmax*.
- **window-spec – TS_PARTIAL_AUTOCORRELATION** is an OLAP function requiring an **OVER ()** clause.

Usage

This function returns an outlier-free time series. **TS_PARTIAL_AUTOCORRELATION** calls the function **imsls_d_autocorrelation** and **imsls_d_partial_autocorrelation** in the IMSL libraries.

IMSL Mapping

The arguments of **TS_PARTIAL_AUTOCORRELATION** map to the IMSL library functions **imsls_d_autocorrelation** and **imsls_d_partial_autocorrelation** as follows:

```
params = imsls_d_autocorrelation(n_objs, z[], lagmax, 0);
```

```
result = imsls_d_partial_autocorrelation(lagmax, params, 0);
```

- **n_objs –** contains the number of rows in the current window frame.
- **z[] –** contains the value of timeseries_expression for the current window frame.
- **lagmax –** maps to the **TS_PARTIAL_AUTOCORRELATION** argument *lagmax*.

For detailed information on how the IMSL functions **imsls_d_autocorrelation** and imsls_d_partial_autocorrelation perform time series calculations, see *IMSL C Numerical Library User's Guide: Volume 2 of 2 C Stat Library*.

Example

This example shows SQL statement containing the **TS_PARTIAL_AUTOCORRELATION** function and the data values returned by the function. This example uses the example input data table (called *DATASET*) as its input data.

The following SQL statement returns the first element from an array containing partial autocorrelations of data from the *data* column:

```
select ts_partial_autocorrelation(data,1,1) over (order by rownum
rows between unbounded preceding and unbounded following) as res FROM
DATASET
```

Sybase IQ returns 50 rows, each containing the same value:

Table 27. Values Returned from TS_PARTIAL_AUTOCORRELATION

res
0.883453
0.883453
0.883453
0.883453
0.883453
0.883453

res
0.883453
0.883453
0.883453
0.883453
...
0.883453

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7
- *DATASET Example Input Data Table* on page 73

TS_VWAP Function [Aggregate]

VWAP stands for volume-weighted average price. **TS_VWAP** calculates the ratio of the value traded to the total volume traded over a particular time horizon. VWAP is a measure of the average price of a stock over a defined trading horizon. You can use **TS_VWAP** as both a simple and an OLAP-style aggregate function. Unlike the other time series functions, **TS_VWAP** does not call the IMSL libraries.

Syntax 1

```
TS_VWAP (price_expression, volume_expression)
```

Syntax 2

```
TS_VWAP (price_expression, volume_expression)
OVER (window-spec)
```

Licensing Prerequisites

Available only with RAP – The Trading Edition Enterprise.

Parameters

- **price_expression –** a numeric expression specifying the price to be incorporated into a volume-weighted average.

- **volume_expression –** a numeric expression specifying the volume to be used in calculating a volume-weighted average.
- **window-spec –** if used with Syntax 2, **TS_VWAP** is an OLAP function requiring an **OVER ()** clause.

Usage

Sybase IQ calculates **TS_VWAP** using the following formula:

Figure 1: VWAP Calculation

$$Pvwap = \frac{\Sigma j \ Pj \cdot Qj}{\Sigma j Qj}$$

Pvwap = volume weighted average price Pj = price of trade j. Qj = quantity of trade j. j = an individual trade that occurred during the time horizon.

Example

This example shows an input data table, a SQL statement containing the **TS_VWAP** function, and the data values returned by the function. This example uses the following table (called *VWAP_DATASET*) as its input data. The VWAP_DATASET table contains three rows of time series data:

Table 28. Input Data Table VWAP_DATASET

rownum	price	volume
1	1	1
2	2	2
3	5	1

The following SQL statement calculates the volume weighted average price:

```
select ts_vwap(price,volume) over (order by rownum Rows between
unbounded preceding and unbounded following) as res FROM VWAP_DATASET
```

Sybase IQ returns three rows:

Table 29. Values Returned from TS_VWAP

res
2.5
2.5

res
2.5

Standards and Compatibility

- **SQL –** ISO/ANSI SQL compliant
- **Sybase –** not compatible with SQL Anywhere or Adaptive Server Enterprise

See also

- *Aggregate Time Series Forecasting and Analysis Functions* on page 7

DATASET Example Input Data Table

Time series and forecasting analysis function examples use the following table (called *DATASET*) as the input data. The DATASET table contains 50 rows of time series data.

Table 30. Input data table DATASET

rownum	data
1	0.315523
2	0.485859
3	0.676886
4	1.97381
5	2.77555
6	2.73657
7	2.64233
8	4.26118
9	3.13641
10	4.16566
11	2.95952
12	2.14504
13	1.98799
14	0.805859
15	0.833405

rownum	data
16	2.29075
17	1.30045
18	0.467122
19	-0.170107
20	-0.256657
21	-0.382597
22	-0.505511
23	-1.90147
24	-0.981688
25	-1.43116
26	-1.39389
27	-2.34823
28	-2.91122
29	-0.927423
30	-0.044383
31	-0.389648
32	0.545008
33	0.614096
34	0.364668
35	1.16043
36	-0.654063
37	0.616094
38	2.00875
39	1.86696
40	2.80171
41	3.78422
42	4.11499
43	2.77188

rownum	data
44	4.00312
45	4.21298
46	5.00413
47	4.74498
48	4.89621
49	3.93273
50	4.31592

See also

- *TS_ARMA_AR Function [Aggregate]* on page 11
- *TS_ARMA_CONST Function [Aggregate]* on page 13
- *TS_ARMA_MA Function [Aggregate]* on page 15
- *TS_AUTOCORRELATION Function [Aggregate]* on page 17
- *TS_AUTO_UNI_AR Function [Aggregate]* on page 35
- *TS_DIFFERENCE Function [Aggregate]* on page 39
- *TS_LACK_OF_FIT Function [Aggregate]* on page 52
- *TS_LACK_OF_FIT_P Function [Aggregate]* on page 54
- *TS_MAX_ARMA_AR Function [Aggregate]* on page 56
- *TS_MAX_ARMA_CONST Function [Aggregate]* on page 59
- *TS_MAX_ARMA_LIKELIHOOD Function [Aggregate]* on page 61
- *TS_MAX_ARMA_MA Function [Aggregate]* on page 63
- *TS_OUTLIER_IDENTIFICATION Function [Aggregate]* on page 65
- *TS_PARTIAL_AUTOCORRELATION Function [Aggregate]* on page 69

Index