Using ⎕SQL

Contents

Overview

The APL64 ⎕SQL system function provides a structured query language (SQL)-based interface to a Microsoft SQL Server database.

The ⎕SQL system function employs the ADO.Net (active data objects) class for .Net which has superior performance and features compared to previous technologies, such as ADODB, ODBC, MDAC, Jet, etc.

With the (no-cost) availability of Microsoft SQL Server Express database, scalable-to-enterprise-level database software is more affordable and the logical choice for professionally designed application systems.

The object model for ⎕SQL includes methods and properties to easily create a secure, high-performance application system data interface.

Prerequisites:

To use ⎕SQL, the target workstation must have appropriate access to an existing SQL Server database. The SQL Server database software may be locally installed, installed on a local server or cloud based.

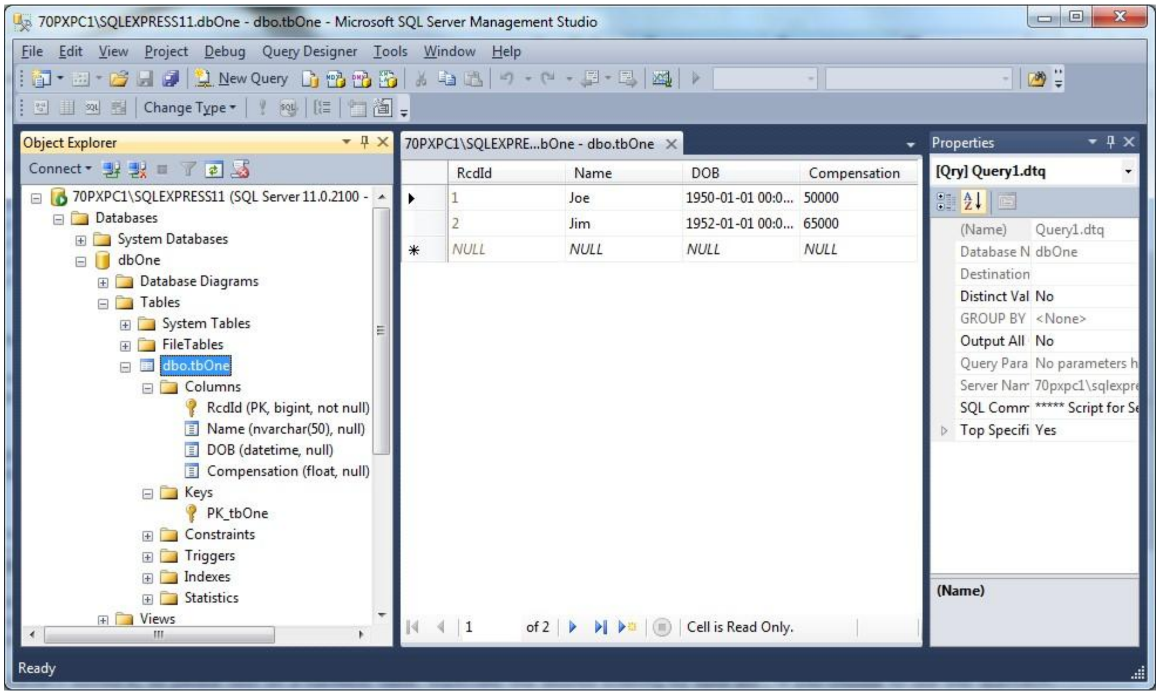
The examples in this document assume that a sample SQL Server database, named ‘dbOne’ is installed, running and accessible to the target workstation. The examples in this document also assume that this sample SQL Server database has a table, named ‘tbOne’, with sample columns and sample data.

The examples in this document are for demonstration purposes, so they may not incorporate enterprise-level security measures, may use a local workstation deployment that is not amenable to sharing data, and use a simplified connection string. For simplicity, some examples do not include all APL64 executable statements which may be necessary to reproduce the illustrated output.

Installation of SQL Server:

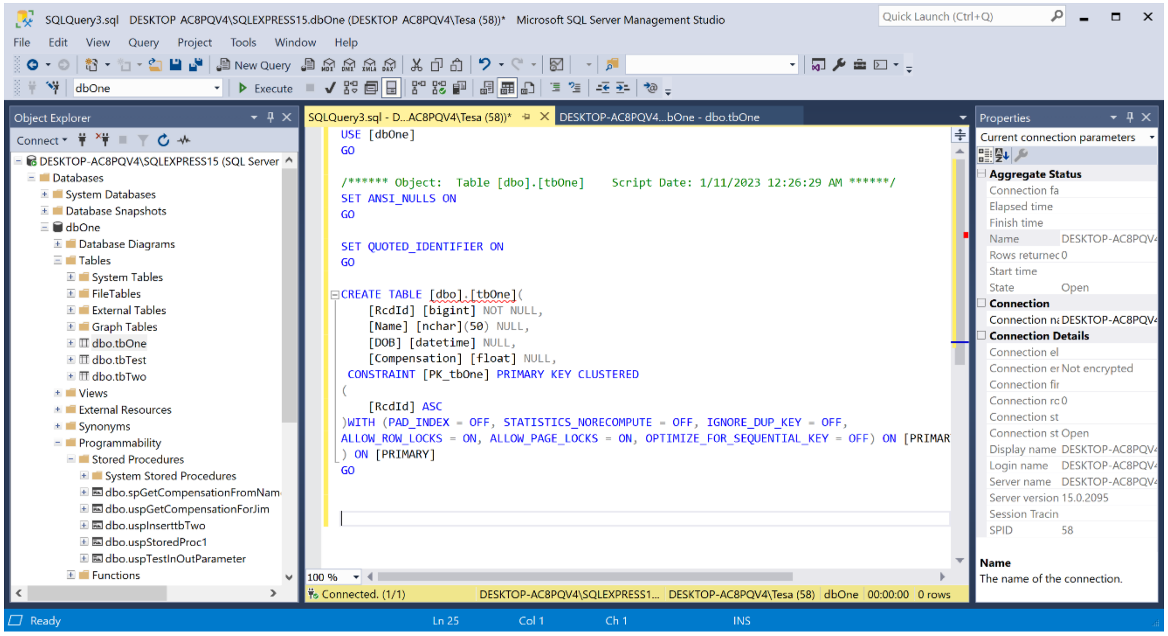
Full installation details of an SQL Server database are beyond the scope of this document. Review the Microsoft SQL Server Express documentation for installation information or Microsoft SQL Server trial version. Consider installing Microsoft SQL Server Management Studio as a tool to establish this instance.

The examples in the remainder of this document are based on the sample database, tables, and values illustrated in the above two documents. The initial state of the sample database as displayed in the SQL Server Management Studio:



	RcdId	Name	DOB	Compensation
▶	1	Joe	1950-01-01 00:0...	50000
	2	Jim	1952-01-01 00:0...	65000
*	NULL	NULL	NULL	NULL

The database name is ‘dbOne’ and the table in the database is ‘tbone’ which can be created using the SQL Server Management Studio:



```
USE [dbOne]
GO

/****** Object:  Table [dbo].[tbOne]    Script Date: 1/11/2023 12:26:29 AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[tbOne](
    [RcdId] [bigint] NOT NULL,
    [Name] [nchar](50) NULL,
    [DOB] [datetime] NULL,
    [Compensation] [float] NULL,
 CONSTRAINT [PK_tbOne] PRIMARY KEY CLUSTERED
(
    [RcdId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMAR
) ON [PRIMARY]
GO
```

Insert the tbOne table sample data:

```
insert into tbOne values (1,'Joe','1960-01-01',50000);
insert into tbOne values (2,'Jim','1961-01-01',60000);
```

Create the sample stored procedure:

```
CREATE PROCEDURE [dbo].[spGetCompensationFromName]
-- Add the parameters for the stored procedure here
    @NameMatch nvarchar(45),
    @COMP float OUTPUT
```

```
AS
BEGIN
       -- SET NOCOUNT ON added to prevent extra result sets from
       -- interfering with SELECT statements.
       SET NOCOUNT ON;

    -- Insert statements for procedure here
       SELECT @COMP = Compensation from tbOne WHERE Name =
@NameMatch
       Select @COMP
END
```

Display Summary Documentation:

⎕SQL Actions

SQL actions operate on either the ⎕SQL object or an ⎕SQL instance. The ⎕sql object is a container for all ⎕sql instances in an APL64 instance. An ⎕sql instance is associated with a specific Sql database accessible to the target workstation user.

The left argument of the APL64 ⎕sql system function determines if the ⎕sql action applies to the ⎕sql object ('#' left argument) or ⎕sql instance (name of ⎕sql instance).

The name of an ⎕sql instance can be specified by:

- An APL64 text expression, e.g. 'myinstance', «myinstance»
- An APL64 variable with value equal to an ⎕sql instance name
- Elided, if the ⎕SqlSelf value is an ⎕sql instance name

⎕sql action names are not case sensitive. ⎕sql instance names are case sensitive

BeginTransaction

This action applies to an ⎕sql instance. The BeginTransaction action, for the Sql instance named in the left argument, indicates that subsequent ⎕sql actions will be incorporated into an SQL transaction. An SQL transaction may contain multiple ⎕sql actions. An SQL transaction can either be committed or rolled back. The BeginTransaction action has no result.

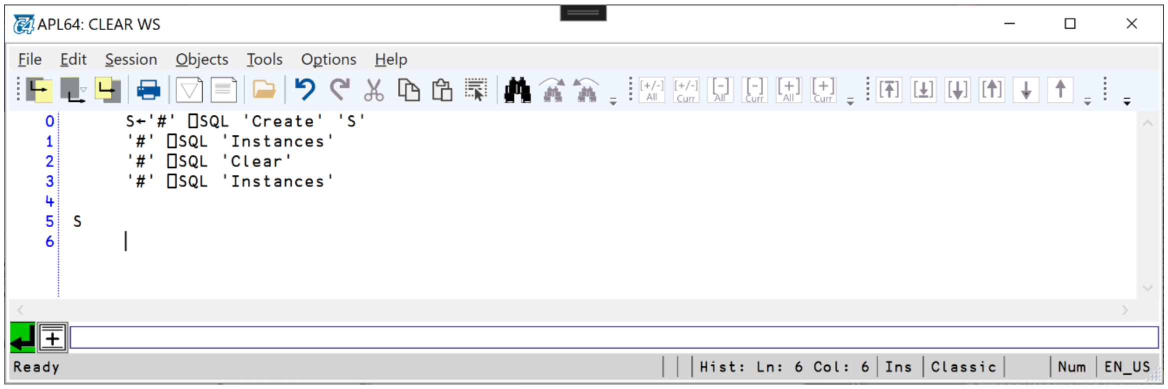
SQL transactions are used to combine multiple SQL statements, possibly affecting multiple tables in an SQL database, so that they are either all performed or none are performed.

```
S←'#' ⎕SQL 'Create' 'S'
S ⎕SQL 'BeginTransaction'
```

Clear

This action is performed on the ⎕SQL object. The Clear action closes the Sql database connection associated with any ⎕SQL instance and deletes all ⎕SQL instances in the current APL64 instance. The Clear action has no result.

```
S←'#' ⎕SQL 'Create' 'S'
 '#' ⎕SQL 'Instances'
 '#' ⎕SQL 'Clear'
 '#' ⎕SQL 'Instances'
```



APL64: CLEAR WS

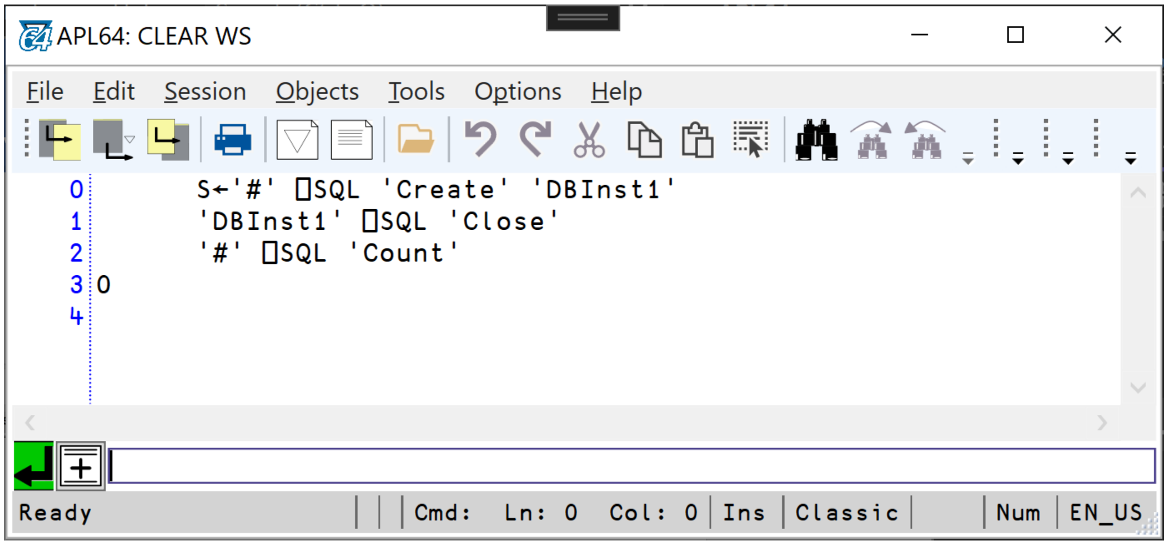
```
      S←'#' ⎕SQL 'Create' 'S'
      '#' ⎕SQL 'Instances'
      '#' ⎕SQL 'Clear'
      '#' ⎕SQL 'Instances'

S
```

Ready | Hist: Ln: 6 Col: 6 | Ins | Classic | Num | EN_US

Close

The close action is performed on an ⎕SQL instance. The close action will close an existing connection and delete the associated ⎕SQL instance.

```
S←'#' ⎕SQL 'Create' 'DBInst1'
'DBInst1' ⎕SQL 'Close'
'#' ⎕SQL 'Count'
```



APL64: CLEAR WS

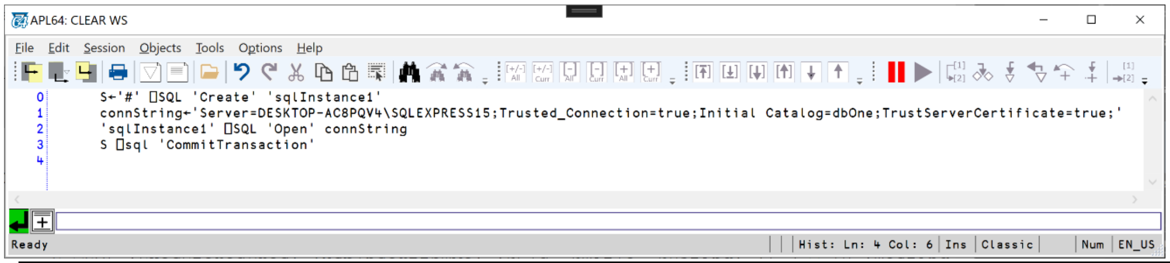
```
      S←'#' ⎕SQL 'Create' 'DBInst1'
      'DBInst1' ⎕SQL 'Close'
      '#' ⎕SQL 'Count'
0
```

Ready | Cmd: Ln: 0 Col: 0 | Ins | Classic | Num | EN_US

CommitTransaction

This action applies to an ⎕SQL instance. The CommitTransaction action causes the effects of the current transaction to be applied to the connected Microsoft Sql Server database. The CommitTransaction action has no effect if the BeginTransaction action has not occurred or if the RollbackTransaction action has occurred. Prior to committal or rollback of a transaction, the effects of the existing SQL statements in the transaction are pending effects in the Sql database. The CommitTransaction action has no result.

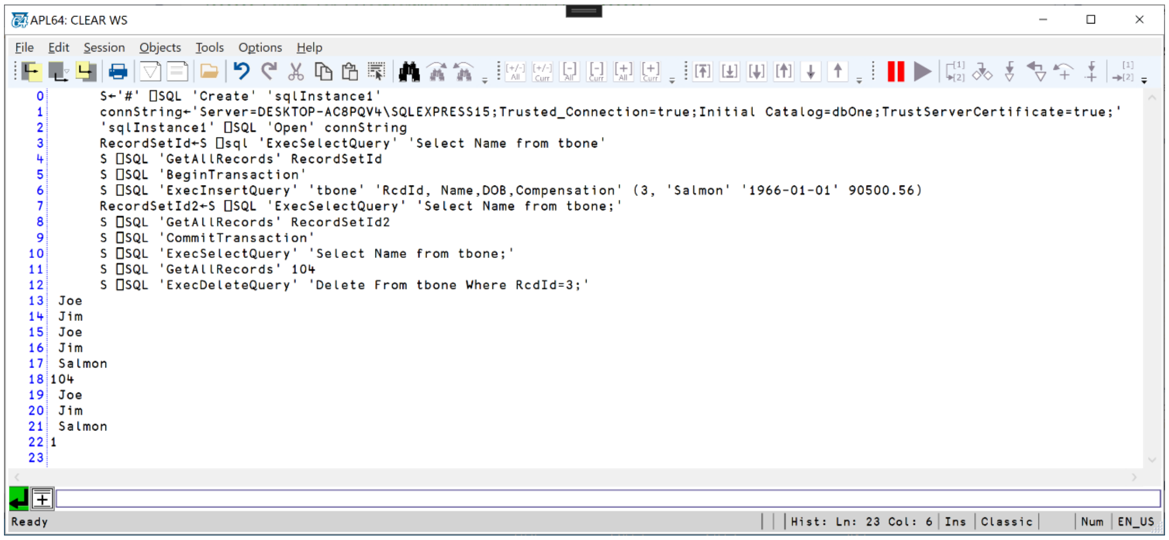
```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
S ⎕sql 'CommitTransaction'
```



```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
'sqlInstance1' ⎕SQL 'Open' connString
S ⎕sql 'CommitTransaction'
```

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
RecordSetId←S ⎕sql 'ExecSelectQuery' 'Select Name from tbone'
S ⎕SQL 'GetAllRecords' RecordSetId
S ⎕SQL 'BeginTransaction'
S ⎕SQL 'ExecInsertQuery' 'tbone' 'RcdId, Name,DOB,Compensation' (3, 'Salmon' '1966-01-01'
90500.56)
RecordSetId2←S ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
S ⎕SQL 'GetAllRecords' RecordSetId2
S ⎕SQL 'CommitTransaction'
S ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
S ⎕SQL 'GetAllRecords' 104
S ⎕SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
```

Note that the records in the RecordSetId2 data set include the pending effects of the ExecInsertQuery action and the records in the #104 data set include the effects of the committed ExecInsertQuery action:

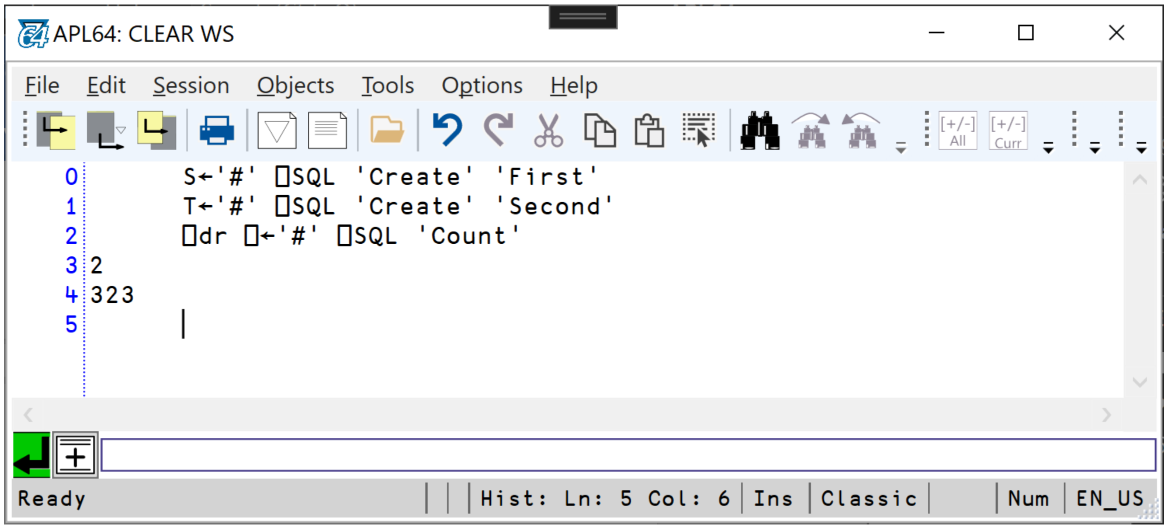


```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      RecordSetId←S ⎕sql 'ExecSelectQuery' 'Select Name from tbone'
      S ⎕SQL 'GetAllRecords' RecordSetId
      S ⎕SQL 'BeginTransaction'
      S ⎕SQL 'ExecInsertQuery' 'tbone' 'RcdId, Name,DOB,Compensation' (3, 'Salmon' '1966-01-01' 90500.56)
      RecordSetId2←S ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
      S ⎕SQL 'GetAllRecords' RecordSetId2
      S ⎕SQL 'CommitTransaction'
      S ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
      S ⎕SQL 'GetAllRecords' 104
      S ⎕SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
 Joe
 Jim
 Joe
 Jim
 Salmon
104
 Joe
 Jim
 Salmon
1
```

Count

This action is performed on the ⎕SQL object. The Create action will create an ⎕SQL instance with a user-provided name in the right argument. Multiple ⎕SQL instances are possible in the same APL64 instance, so that multiple Sql Server databases may be conveniently accessed.

```
S←'#' ⎕SQL 'Create' 'First'
T←'#' ⎕SQL 'Create' 'Second'
⎕dr ⎕←'#' ⎕SQL 'Count'
```



```
      S←'#' ⎕SQL 'Create' 'First'
      T←'#' ⎕SQL 'Create' 'Second'
      ⎕dr ⎕←'#' ⎕SQL 'Count'
2
323
```

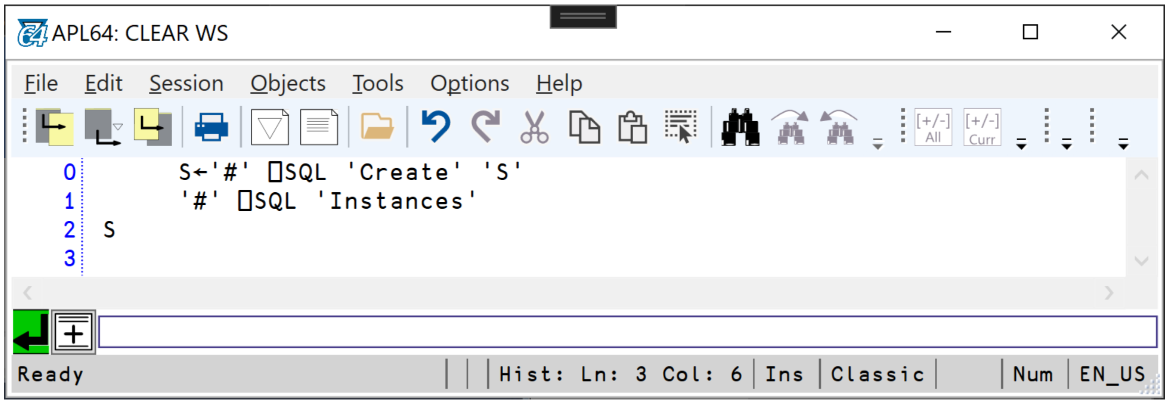
Create

This action is performed on the ⎕SQL object. The Create action will create an ⎕SQL instance with a user-provided name in the right argument. Multiple ⎕SQL instances are possible in the same APL64 instance, so that multiple Sql Server databases may be conveniently accessed.

The Create action will not fail if the named instance already exists. The Create action will close and delete a pre-existing instance with the same name. The Create action for a particular Sql

Server database is generally used once in an APL64 instance. The Create action does not open a connection to an Sql Server database, use the Open action for that purpose. The result of a successful 'Create' action is a text vector containing the ⎕SQL instance name.

```
S←'#' ⎕SQL 'Create' 'S'
'#' ⎕SQL 'Instances'
```


APL64: CLEAR WS

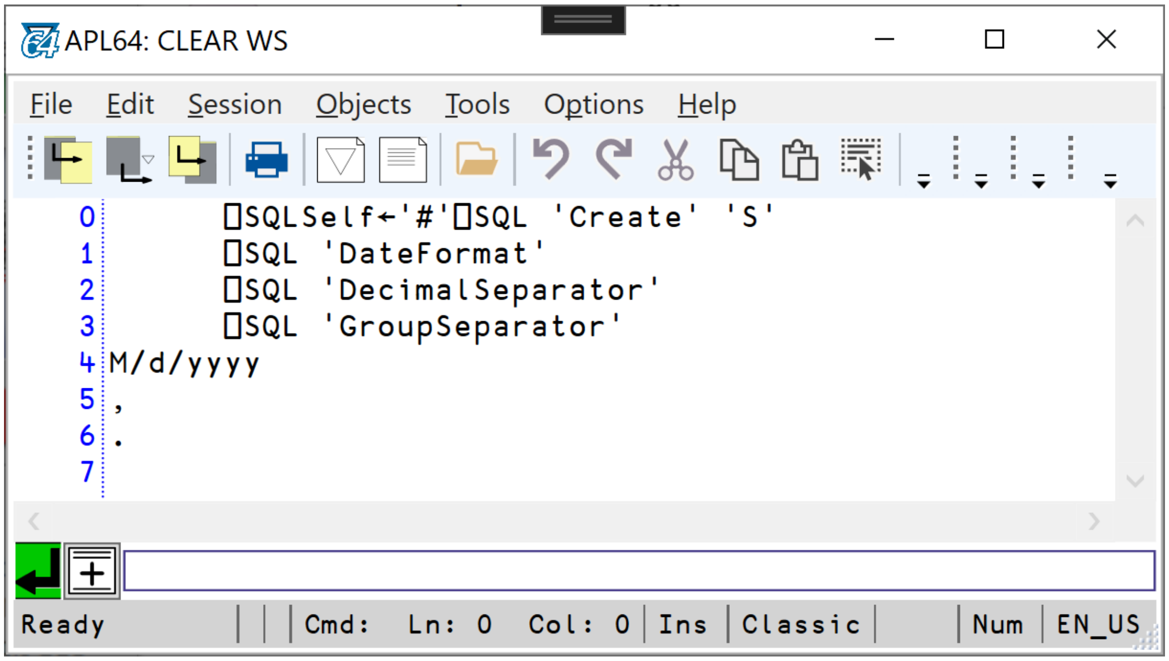
```
      S←'#' ⎕SQL 'Create' 'S'
      '#' ⎕SQL 'Instances'
S
```

Ready | Hist: Ln: 3 Col: 6 | Ins | Classic | Num | EN_US

DateFormat

This action returns the current .Net Short Date Pattern text applicable to the specified ⎕SQL instance.

```
⎕SQLSelf←'#'⎕SQL 'Create' 'S'
⎕SQL 'DateFormat'
⎕SQL 'DecimalSeparator'
⎕SQL 'GroupSeparator'
```


APL64: CLEAR WS

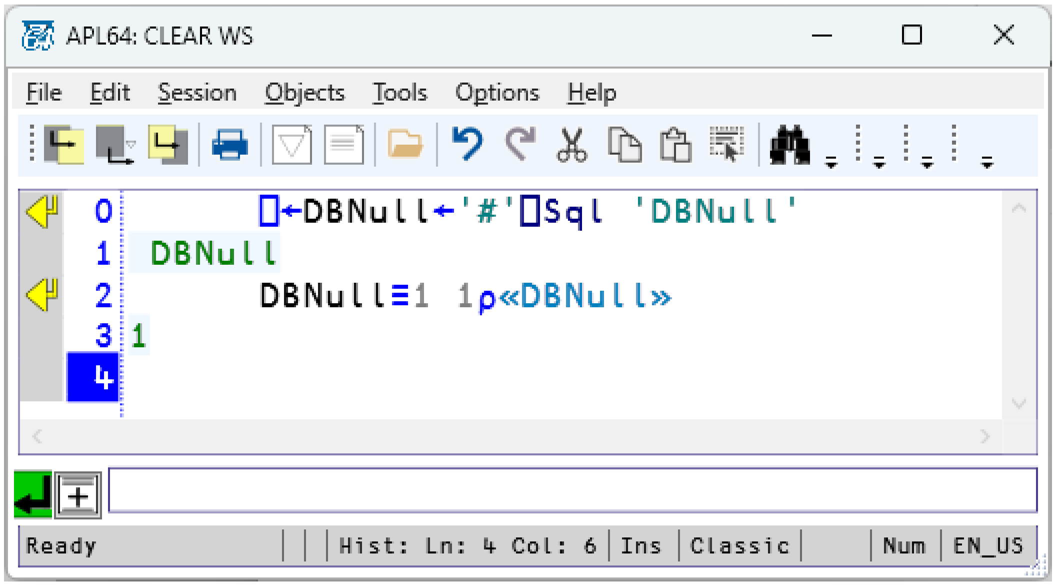
```
      ⎕SQLSelf←'#'⎕SQL 'Create' 'S'
      ⎕SQL 'DateFormat'
      ⎕SQL 'DecimalSeparator'
      ⎕SQL 'GroupSeparator'
M/d/yyyy
,
.
```

Ready | Cmd: Ln: 0 Col: 0 | Ins | Classic | Num | EN_US

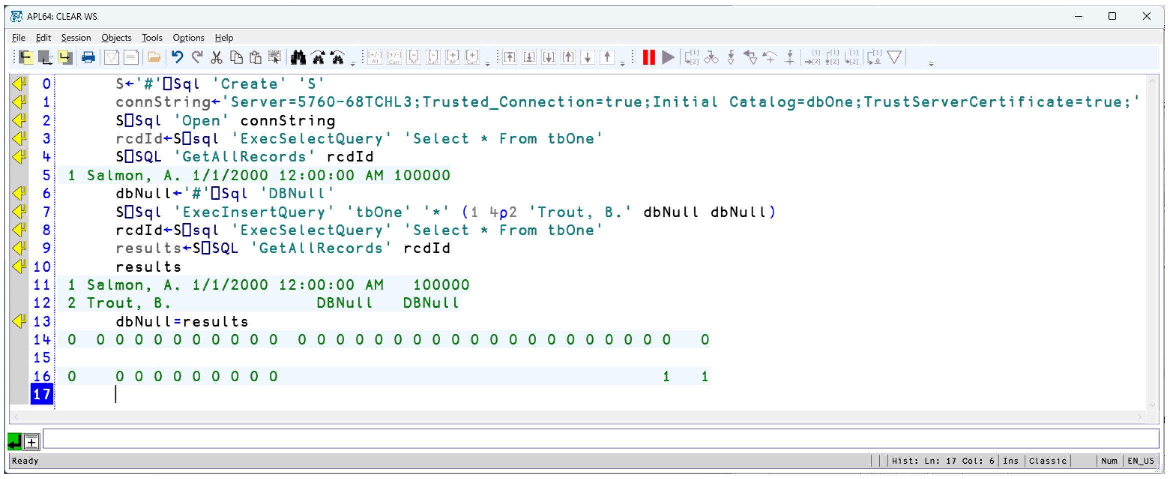
DBNull

This action returns a value which may be used to specify null values for the ExecInsertQuery and ExecStoredProc ⎕SQL instance actions. This value will also be used as the APL64 value associated with a DBNull value returned from a GetRecord, GetAllRecords, or ExecStoredProc action.

```
⎕←DBNull←'#'⎕Sql 'DBNull'
DBNull≡1 1ρ«DBNull»
```



```
      ⎕←DBNull←'#'⎕Sql 'DBNull'
 DBNull
      DBNull≡1 1ρ«DBNull»
1
```



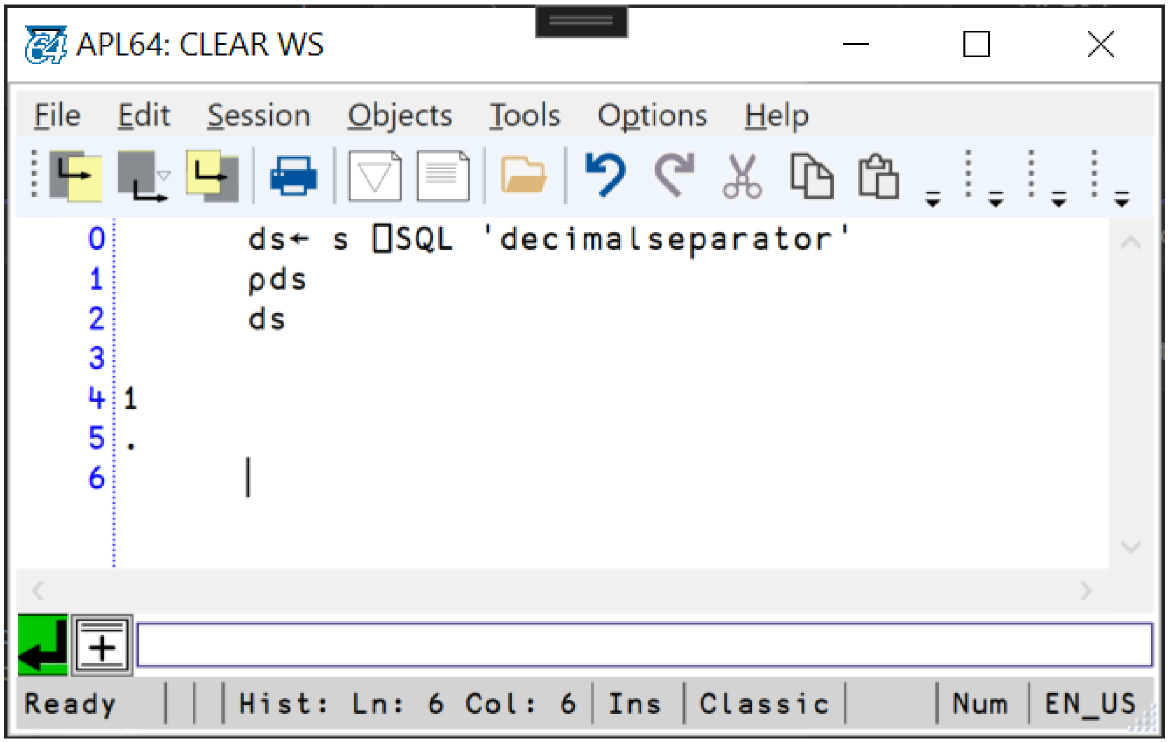
```
      S←'#'⎕Sql 'Create' 'S'
      connString←'Server=5760-68TCHL3;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      S⎕Sql 'Open' connString
      rcdId←S⎕sql 'ExecSelectQuery' 'Select * From tbOne'
      S⎕SQL 'GetAllRecords' rcdId
1 Salmon, A. 1/1/2000 12:00:00 AM 100000
      dbNull←'#'⎕Sql 'DBNull'
      S⎕Sql 'ExecInsertQuery' 'tbOne' '*' (1 4ρ2 'Trout, B.' dbNull dbNull)
      rcdId←S⎕sql 'ExecSelectQuery' 'Select * From tbOne'
      results←S⎕SQL 'GetAllRecords' rcdId
      results
1 Salmon, A. 1/1/2000 12:00:00 AM   100000
2 Trout, B.                DBNull   DBNull
      dbNull=results
0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0

0    0 0 0 0 0 0 0 0 0                                       1   1
```

DecimalSeparator

This action returns the current .Net Number Decimal Separator text applicable to the specified ⎕SQL instance.

```
s←'#' ⎕SQL 'Create' 'sInstanceName'
ds← s ⎕SQL 'decimalseparator'
```

```
⍴ds
ds
```



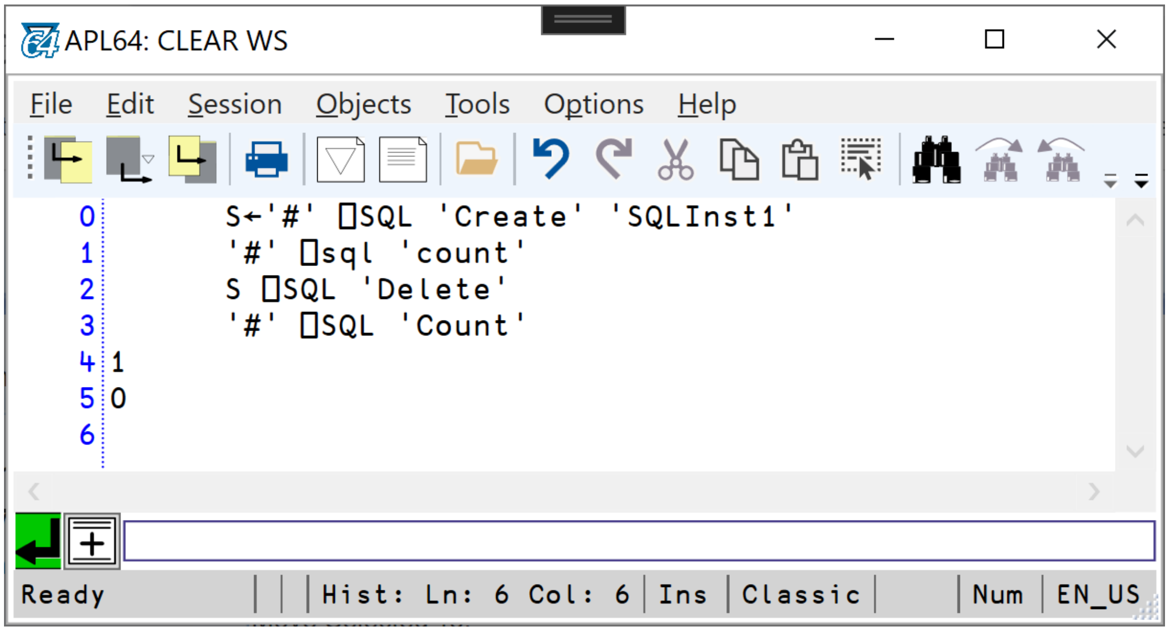
APL64: CLEAR WS

```
      ds← s ⎕SQL 'decimalseparator'
      ⍴ds
      ds

1
.
```

Delete

The Delete action is performed on an ⎕SQL instance. The close action will close an existing connection and delete the associated ⎕SQL instance.

```
S←'#' ⎕SQL 'Create' 'SQLInst1'
'#' ⎕sql 'count'
S ⎕SQL 'Delete'
'#' ⎕SQL 'Count'
```


APL64: CLEAR WS
File Edit Session Objects Tools Options Help

```
      S←'#' ⎕SQL 'Create' 'SQLInst1'
      '#' ⎕sql 'count'
      S ⎕SQL 'Delete'
      '#' ⎕SQL 'Count'
1
0
```

Ready | Hist: Ln: 6 Col: 6 | Ins | Classic | Num | EN_US

Exec

This action applies to an ⎕SQL instance. The ⎕SQL Exec action returns the number of database rows, if any, affected by the executed SQL statement provided by the right argument. The ⎕SQL Exec action cannot be used to return results of a Select SQL statement.

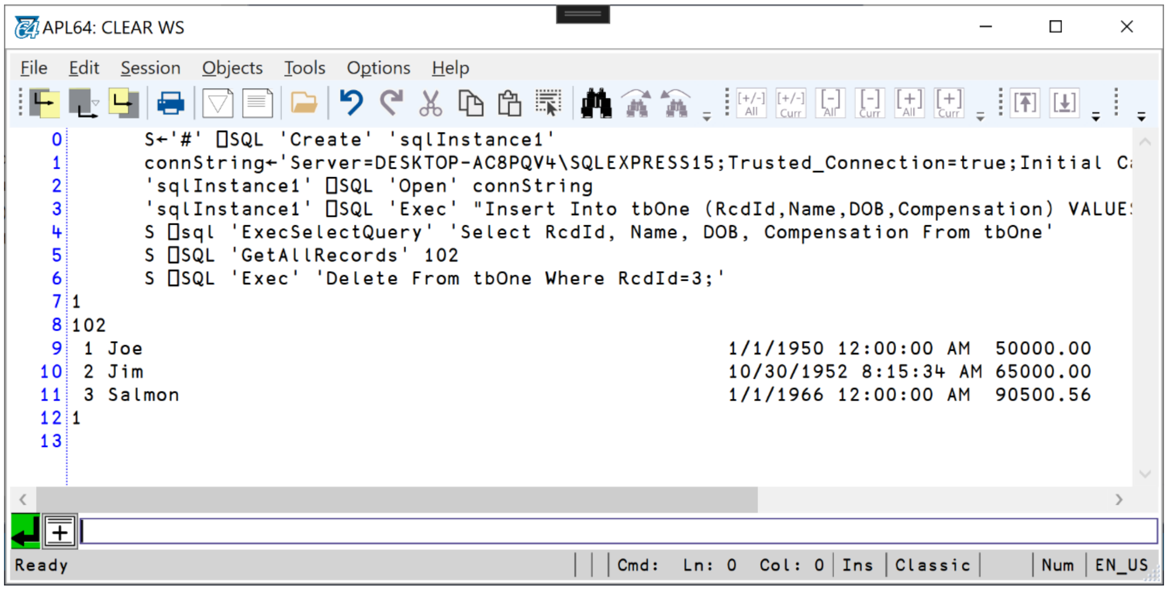
Syntax: [sqlInstance] ⎕SQL 'Exec' cmd [subValue1] [subValue2] …

APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the expression.

Example: ⎕SQL 'Exec' 'Delete From tbOne Where RcdId={0};' 3

Example:

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
'sqlInstance1' ⎕SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUES
(3,'Salmon','1966-01-01',90500.56);"
S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
S ⎕SQL 'GetAllRecords' 102
S ⎕SQL 'Exec' 'Delete From tbOne Where RcdId=3;'
```


APL64: CLEAR WS
File Edit Session Objects Tools Options Help
Ready
Cmd: Ln: 0 Col: 0 Ins Classic Num EN_US

```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial C
      'sqlInstance1' ⎕SQL 'Open' connString
      'sqlInstance1' ⎕SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUE
      S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
      S ⎕SQL 'GetAllRecords' 102
      S ⎕SQL 'Exec' 'Delete From tbOne Where RcdId=3;'
1
102
 1 Joe                                                    1/1/1950 12:00:00 AM  50000.00
 2 Jim                                                    10/30/1952 8:15:34 AM 65000.00
 3 Salmon                                                 1/1/1966 12:00:00 AM  90500.56
1
```

ExecDeleteQuery

Synonym: ExecDelete

Syntax: [sqlInstance] ⎕SQL 'ExecDelete' cmd [subValue1] [subValue2] …

This action applies to an ⎕SQL instance. The right argument is the text of the SQL Delete statement. The result of the ⎕SQL ExecDeleteQuery is the number of records affected, if any.

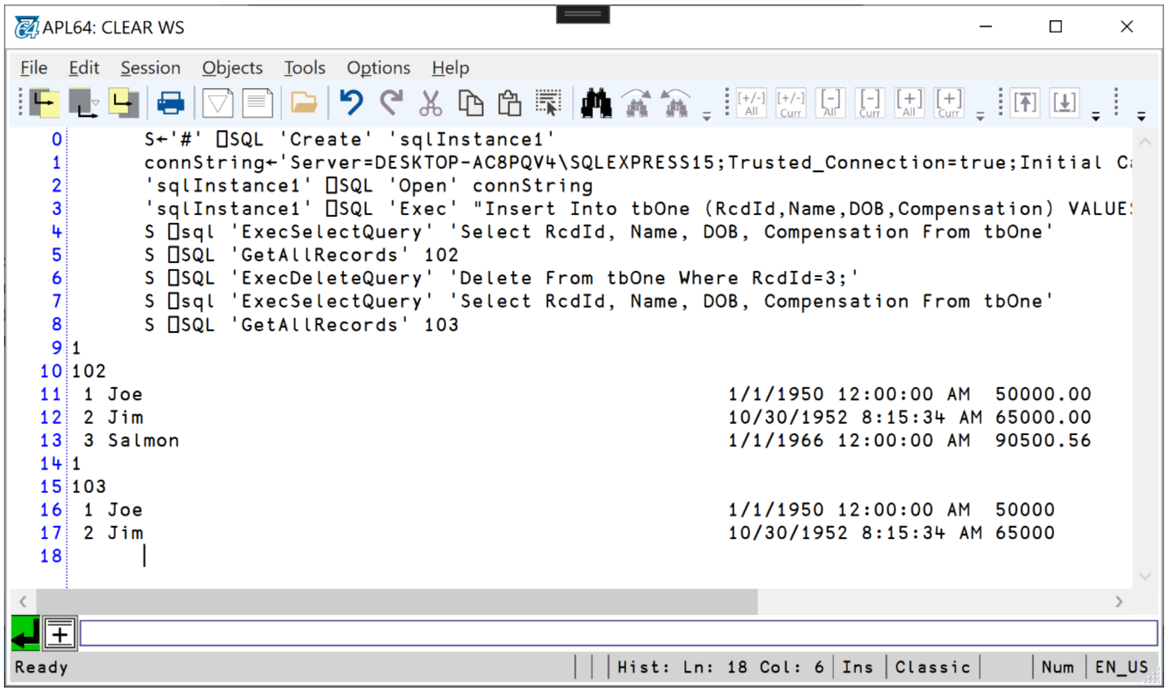
APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the expression.

Example: ⎕SQL 'ExecDelete' 'Delete From tbOne Where RcdId={0};' 3

Example:

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
'sqlInstance1' ⎕SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUES
(3,'Salmon','1966-01-01',90500.56);"
S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
S ⎕SQL 'GetAllRecords' 102
S ⎕SQL 'ExecDeleteQuery' 'Delete From tbOne Where RcdId=3;'
```

```
S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
S ⎕SQL 'GetAllRecords' 103
```


APL64: CLEAR WS
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 18 Col: 6 Ins Classic Num EN_US

```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial C
      'sqlInstance1' ⎕SQL 'Open' connString
      'sqlInstance1' ⎕SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUE
      S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
      S ⎕SQL 'GetAllRecords' 102
      S ⎕SQL 'ExecDeleteQuery' 'Delete From tbOne Where RcdId=3;'
      S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
      S ⎕SQL 'GetAllRecords' 103
1
102
 1 Joe                                                 1/1/1950 12:00:00 AM  50000.00
 2 Jim                                                 10/30/1952 8:15:34 AM 65000.00
 3 Salmon                                              1/1/1966 12:00:00 AM  90500.56
1
103
 1 Joe                                                 1/1/1950 12:00:00 AM  50000
 2 Jim                                                 10/30/1952 8:15:34 AM 65000
```

ExecInsertQuery

Synonym: ExecInsert

This action applies to an ⎕SQL instance.
Right arguments:

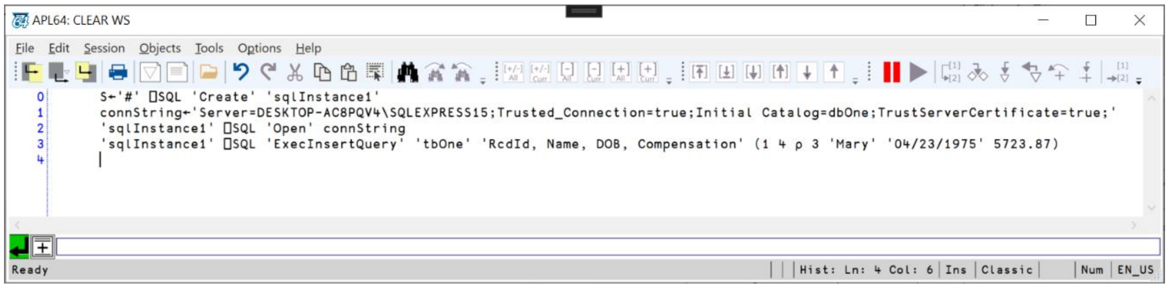
1. Sql table name
2. Sql column names: Comma delimited text with selected Sql table column names, or '*' to indicate all Sql table column names.
3. APL64 data: A matrix containing the APL64 values to be inserted. One row for each record to be inserted.

The order of the Sql column names and APL64 data matrix columns must be conformable.

Example:

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

 'sqlInstance1' ⎕SQL 'Open' connString
```

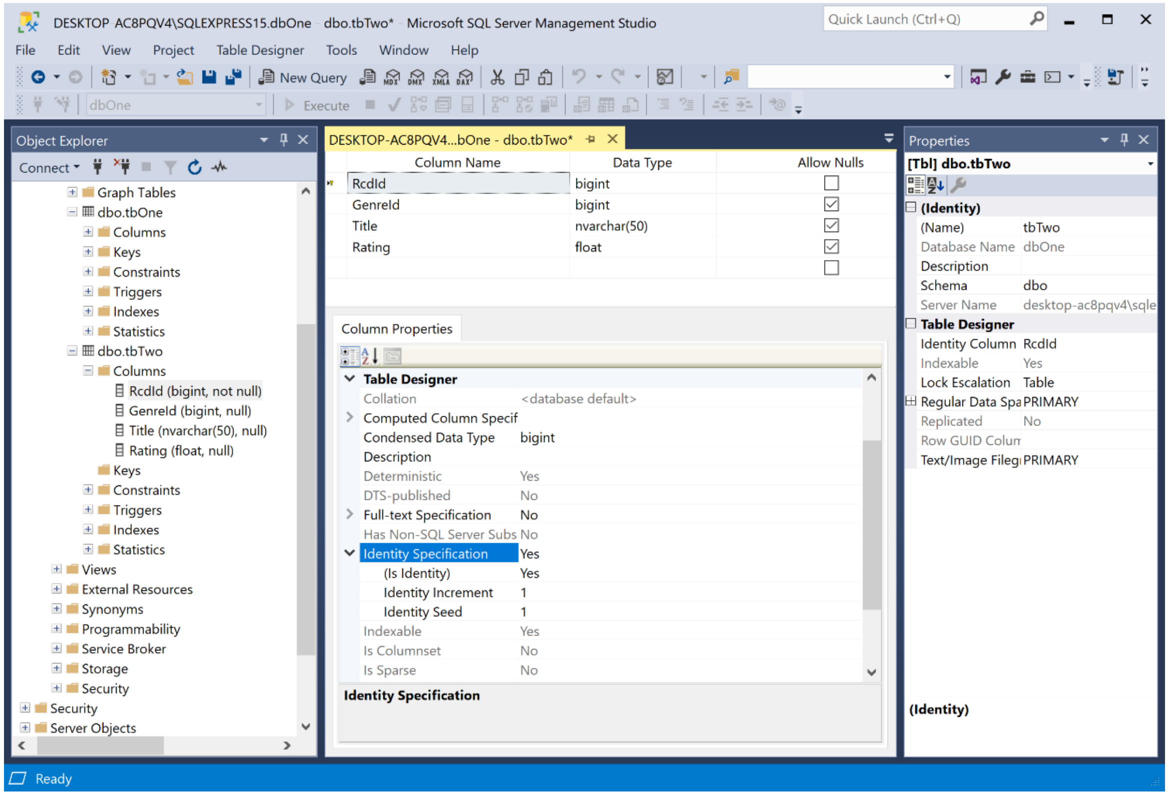
```
'sqlInstance1' ⎕SQL 'ExecInsertQuery' 'tbOne' 'RcdId, Name, DOB, Compensation' (1 4 ρ 3
'Mary' '04/23/1975' 5723.87)
```



```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
'sqlInstance1' ⎕SQL 'Open' connString
'sqlInstance1' ⎕SQL 'ExecInsertQuery' 'tbOne' 'RcdId, Name, DOB, Compensation' (1 4 ρ 3 'Mary' '04/23/1975' 5723.87)
```

The argument syntax for this method is:

- Argument #1: table name: The name of the database table into which records will be inserted
- Argument #2: fields: Either a string consisting of an asterisk (signifying all columns) or a comma-separated string of table column names for which values are provided in the records to be inserted.
- Argument #3: data: An array of APL64 data for the selected fields with one row for each record to be inserted and columns corresponding to each of the fields specified in Argument #2. If Argument #2 is '*' then Argument #3 must contain a value for every column in table Argument #1 that is not auto-generated.



DESKTOP-AC8PQV4\SQLEXPRESS15.dbOne - dbo.tbTwo* - Microsoft SQL Server Management Studio

```
dbo.tbOne
  Columns
  Keys
  Constraints
  Triggers
  Indexes
  Statistics
dbo.tbTwo
  Columns
    RcdId (bigint, not null)
    GenreId (bigint, null)
    Title (nvarchar(50), null)
    Rating (float, null)
  Keys
  Constraints
  Triggers
  Indexes
  Statistics
```

Column Name	Data Type	Allow Nulls
RcdId	bigint	☐
GenreId	bigint	✓
Title	nvarchar(50)	✓
Rating	float	✓

Column Properties

Table Designer	
Collation	<database default>
Computed Column Specif	
Condensed Data Type	bigint
Description	
Deterministic	Yes
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subs	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1
Indexable	Yes
Is Columnset	No
Is Sparse	No

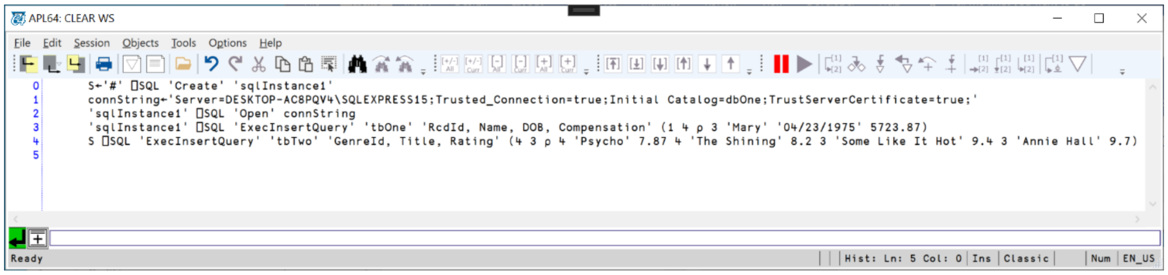
Properties: [Tbl] dbo.tbTwo

(Identity)	
(Name)	tbTwo
Database Name	dbOne
Description	
Schema	dbo
Server Name	desktop-ac8pqv4\sqle
Table Designer	
Identity Column	RcdId
Indexable	Yes
Lock Escalation	Table
Regular Data Spa	PRIMARY
Replicated	No
Row GUID Colum	
Text/Image Fileg	PRIMARY

The examples demonstrate using the ExecInsertQuery for a table without an Identity-type Primary Key field (tbOne) and a table with an Identity-type Primary Key field (tbTwo). The examples also demonstrate executing an insert statement specifying fields and an asterisk which indicates all fields.

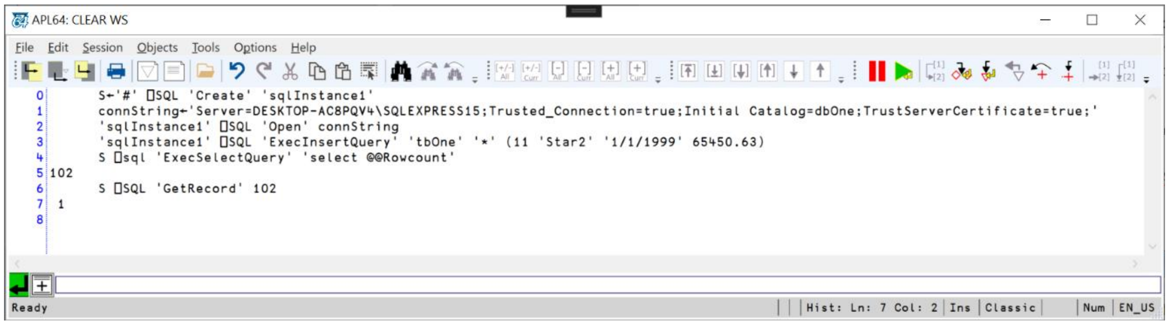
'tbOne' does not contain such a field (Identity Column), so an Insert SQL statement should use the Exec or ExecuteSelectQuery methods. In the example below an additional table, 'tbTwo' was defined in the database which has an Identity field: RcdId.

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'

 'sqlInstance1' ⎕SQL 'Open' connString
 'sqlInstance1' ⎕SQL 'ExecInsertQuery' 'tbOne' 'RcdId, Name, DOB, Compensation' (1 4 ρ 3 'Mary' '04/23/1975' 5723.87)

S ⎕SQL 'ExecInsertQuery' 'tbTwo' 'GenreId, Title, Rating' (4 3 ρ 4 'Psycho' 7.87 4 'The Shining' 8.2 3 'Some Like It Hot' 9.4 3 'Annie Hall' 9.7)
```

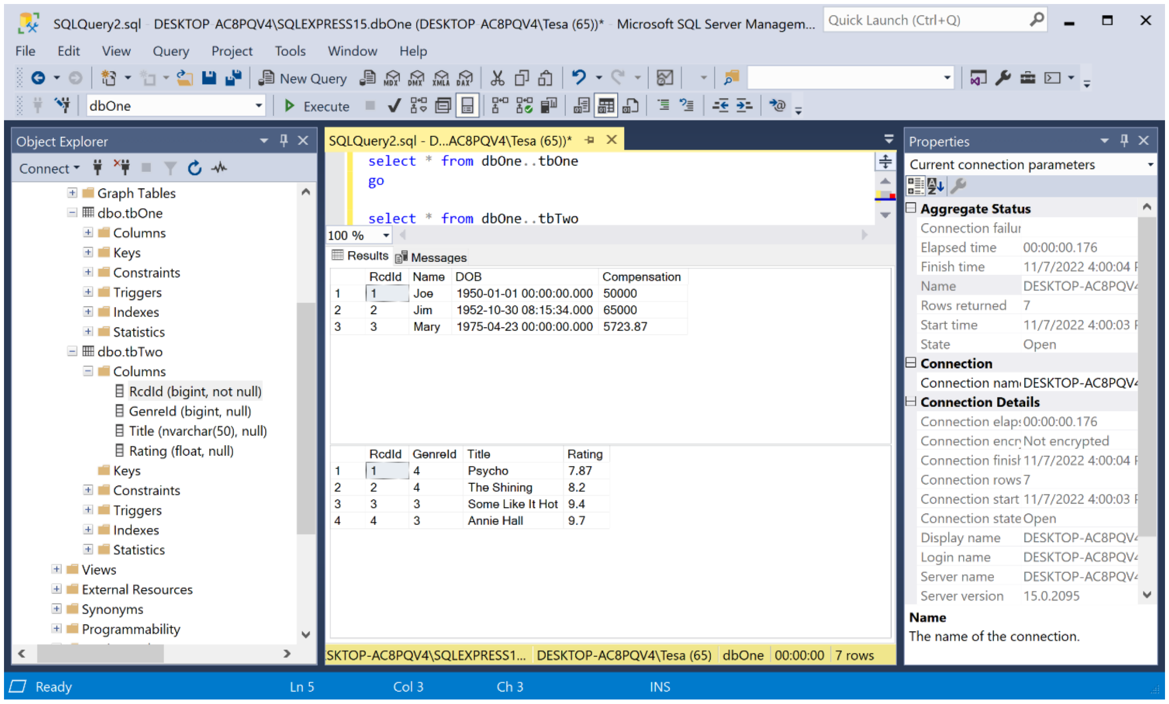


```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      'sqlInstance1' ⎕SQL 'ExecInsertQuery' 'tbOne' 'RcdId, Name, DOB, Compensation' (1 4 ρ 3 'Mary' '04/23/1975' 5723.87)
      S ⎕SQL 'ExecInsertQuery' 'tbTwo' 'GenreId, Title, Rating' (4 3 ρ 4 'Psycho' 7.87 4 'The Shining' 8.2 3 'Some Like It Hot' 9.4 3 'Annie Hall' 9.7)
```

Using Microsoft SQL Server Management Studio to query the contents of both tables tbOne and tbTwo displays the result of the ExecInsertQuery statements.



```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      'sqlInstance1' ⎕SQL 'ExecInsertQuery' 'tbOne' '*' (11 'Star2' '1/1/1999' 65450.63)
      S ⎕sql 'ExecSelectQuery' 'select @@Rowcount'
102
      S ⎕SQL 'GetRecord' 102
 1
```


SQLQuery2.sql - DESKTOP-AC8PQV4\SQLEXPRESS15.dbOne (DESKTOP-AC8PQV4\Tesa (65))* - Microsoft SQL Server Managem...

Object Explorer

- Graph Tables
- dbo.tbOne
  - Columns
  - Keys
  - Constraints
  - Triggers
  - Indexes
  - Statistics
- dbo.tbTwo
  - Columns
    - RcdId (bigint, not null)
    - GenreId (bigint, null)
    - Title (nvarchar(50), null)
    - Rating (float, null)
  - Keys
  - Constraints
  - Triggers
  - Indexes
  - Statistics
- Views
- External Resources
- Synonyms
- Programmability

```
select * from dbOne..tbOne
go

select * from dbOne..tbTwo
```

	RcdId	Name	DOB	Compensation
1	1	Joe	1950-01-01 00:00:00.000	50000
2	2	Jim	1952-10-30 08:15:34.000	65000
3	3	Mary	1975-04-23 00:00:00.000	5723.87

	RcdId	GenreId	Title	Rating
1	1	4	Psycho	7.87
2	2	4	The Shining	8.2
3	3	3	Some Like It Hot	9.4
4	4	3	Annie Hall	9.7

SKTOP-AC8PQV4\SQLEXPRESS1... | DESKTOP-AC8PQV4\Tesa (65) | dbOne | 00:00:00 | 7 rows

Properties — Current connection parameters

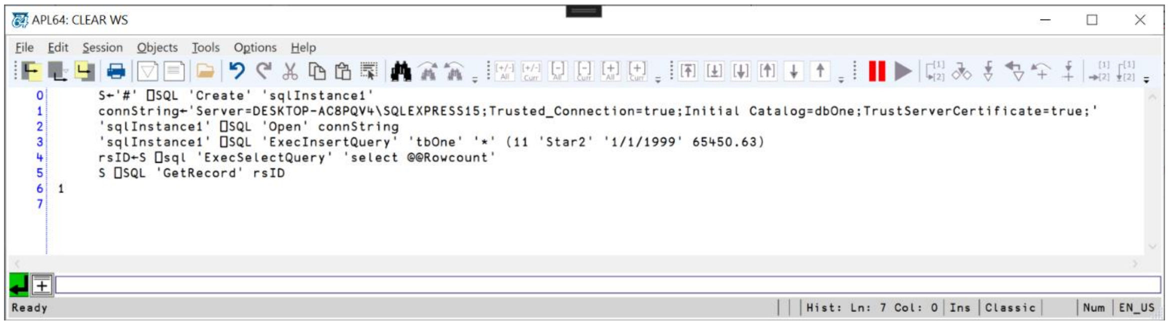
Aggregate Status	
Connection failur	
Elapsed time	00:00:00.176
Finish time	11/7/2022 4:00:04 I
Name	DESKTOP-AC8PQV4
Rows returned	7
Start time	11/7/2022 4:00:03 I
State	Open
Connection	
Connection nam	DESKTOP-AC8PQV4
Connection Details	
Connection elap	00:00:00.176
Connection encr	Not encrypted
Connection finisl	11/7/2022 4:00:04 I
Connection rows	7
Connection start	11/7/2022 4:00:03 I
Connection state	Open
Display name	DESKTOP-AC8PQV4
Login name	DESKTOP-AC8PQV4
Server name	DESKTOP-AC8PQV4
Server version	15.0.2095

Name
The name of the connection.

Ready Ln 5 Col 3 Ch 3 INS

Illustration of insert query with no fields specified ('*'):

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
'sqlInstance1' ⎕SQL 'ExecInsertQuery' 'tbOne' '*' (11 'Star2' '1/1/1999' 65450.63)
rsID←S ⎕sql 'ExecSelectQuery' 'select @@Rowcount'
S ⎕SQL 'GetRecord' rsID
```


APL64: CLEAR WS

```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      'sqlInstance1' ⎕SQL 'ExecInsertQuery' 'tbOne' '*' (11 'Star2' '1/1/1999' 65450.63)
      rsID←S ⎕sql 'ExecSelectQuery' 'select @@Rowcount'
      S ⎕SQL 'GetRecord' rsID
1
```

Ready | Hist: Ln: 7 Col: 0 | Ins | Classic | Num | EN_US

ExecSelectQuery

Synonym: ExecSelect

Syntax: [sqlInstance] ⎕SQL 'ExecSelect' cmd [subValue1] [subValue2] …

The ExecSelectQuery action applies to an ⎕sql instance. The action argument provides the SQL Select statement. Depending on the Sql data type of a column selected, it may be necessary to cast that data column values to a data type which has a representation in APL64.

The ExecSelectQuery does not return the result of the SQL Select statement. The result is an integer indicating the record set containing the result of the SQL Select statement. Use the 'GetRecord' or 'GetAllRecords' actions to access the specified record set.

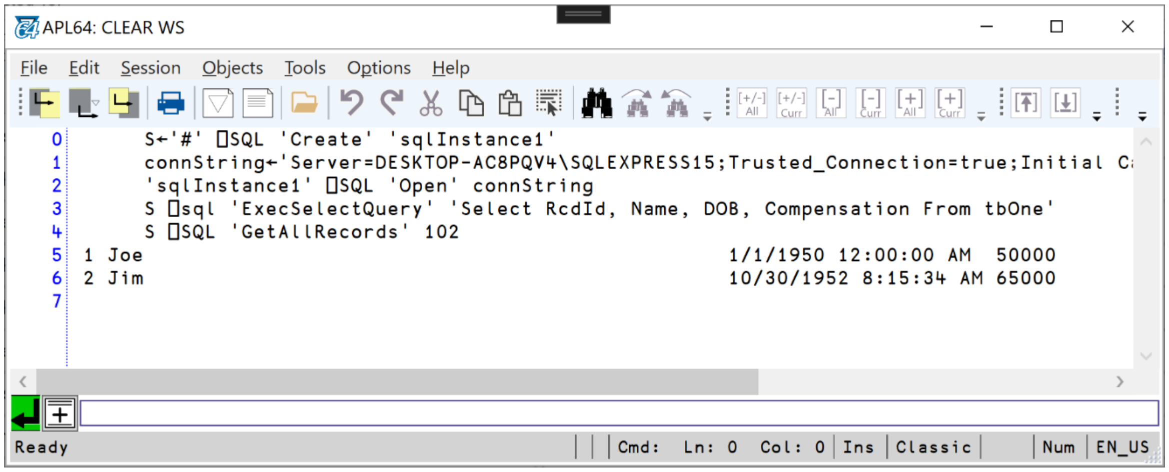
The SQL-format query text argument to the ⎕SQL ExecSelectQuery method, should yield only one record set. A query of the form 'query1;…;queryN', is not supported in APL64 and must be separated into individual queries. As an alternative to separate queries, a stored procedure can be created and executed using the ⎕SQL ExecStoredProc method.

APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the expression.

Example: ⎕SQL 'ExecSelect' 'Select * From tbOne Where RcdId={0};' 3

Example:

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
 S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
 S ⎕SQL 'GetAllRecords' 102
```



```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial C
      'sqlInstance1' ⎕SQL 'Open' connString
      S ⎕sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
      S ⎕SQL 'GetAllRecords' 102
1 Joe                                             1/1/1950 12:00:00 AM  50000
2 Jim                                             10/30/1952 8:15:34 AM 65000
```

Ready Cmd: Ln: 0 Col: 0 Ins Classic Num EN_US

ExecStoredProc

This action can be used to run a stored procedure which has input, output and combined input/output parameters.

The action arguments are:

- An APL text vector (cmdText) containing the stored procedure name
- An APL comma-delimited text vector containing the parameter specifications. A parameter specification contains up to three text elements, each separated by a space:
  - Case-sensitive parameter name (Required)
  - 'In', 'Out' or 'InOut' (Required)
  - APL64-compatible data type [Only for 'Out' parameters]
- An APL rank-1 vector of 'In' and 'InOut' parameter values. The order of the parameter values and the order of the 'In' and 'InOut' parameter names must be conformable.

Not all Sql data types can be represented in APL64, so the data types of stored procedure parameters must be carefully considered. For Sql data types which have no representation in APL64, the stored procedure must cast the parameter values to APL64-compatible data types.

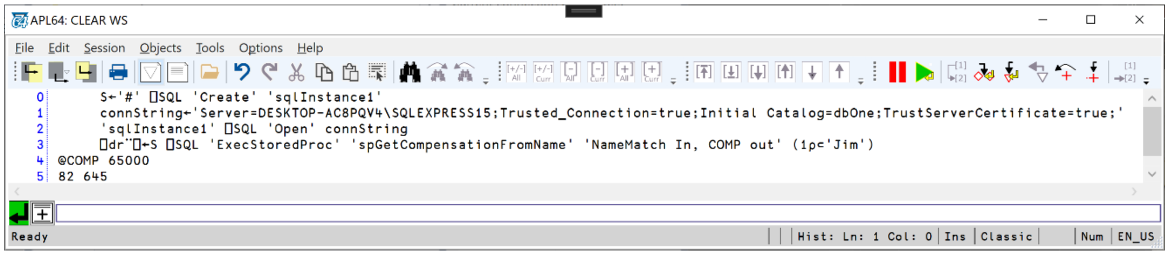
- The Sql data types of In, InOut and Out stored procedure parameters must be compatible with APL64 data types.
- For Out parameters, there are two options to indicate the data type specified in the stored procedure definition:
  - Specify an APL64-compatible data type in the optional third element of the parameter specifications, e.g. 'paramName Out dataType'. If the provided data type text cannot be parsed into a DbDataType, the String type is used.
  - If the optional third element of the parameter specifications is not provided, e.g. 'paramName Out', the system will attempt to obtain the Out parameter's data type from the database schema of the stored procedure. If this attempt fails, the String data type is used.
  - A stored procedure Out parameter should have no 'place holder' in the dataVector.
- The APL64-compatible data types are Boolean, Double, Int32, Char, Varchar and String.

The result is an APL 2-column matrix with one row for each out and in-out parameter:

- Column #1: Parameter name
- Column #2: Parameter value

In and Out parameters with the In parameter specified before the Out parameter:

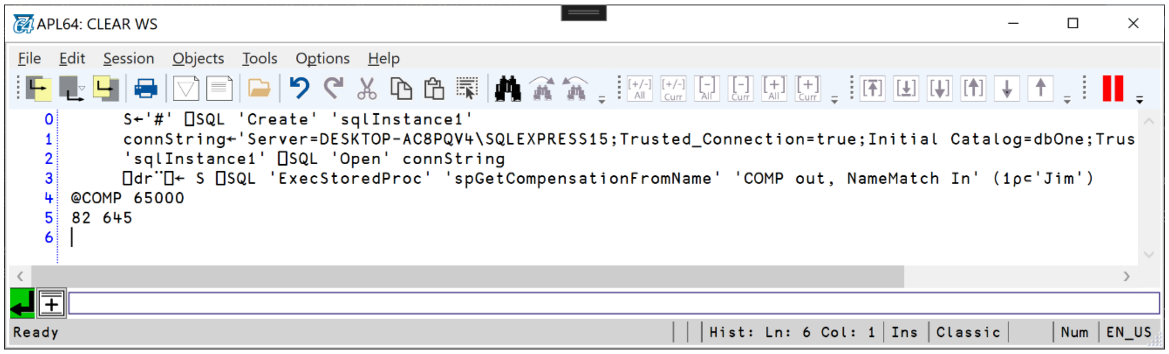
```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
⎕dr¨⎕← S ⎕SQL 'ExecStoredProc' 'spGetCompensationFromName' 'NameMatch In, COMP
Out' (1ρ⊂'Jim')
```



```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      ⎕dr¨⎕←S ⎕SQL 'ExecStoredProc' 'spGetCompensationFromName' 'NameMatch In, COMP out' (1ρ⊂'Jim')
@COMP 65000
82 645
```

In and Out parameters with the Out parameter specified before the In parameter:

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
⎕dr¨⎕← S ⎕SQL 'ExecStoredProc' 'spGetCompensationFromName' 'COMP out, NameMatch In'
(1ρ⊂'Jim')
```



```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;Trus
      'sqlInstance1' ⎕SQL 'Open' connString
      ⎕dr¨⎕← S ⎕SQL 'ExecStoredProc' 'spGetCompensationFromName' 'COMP out, NameMatch In' (1ρ⊂'Jim')
@COMP 65000
82 645
```

The stored procedure definition:

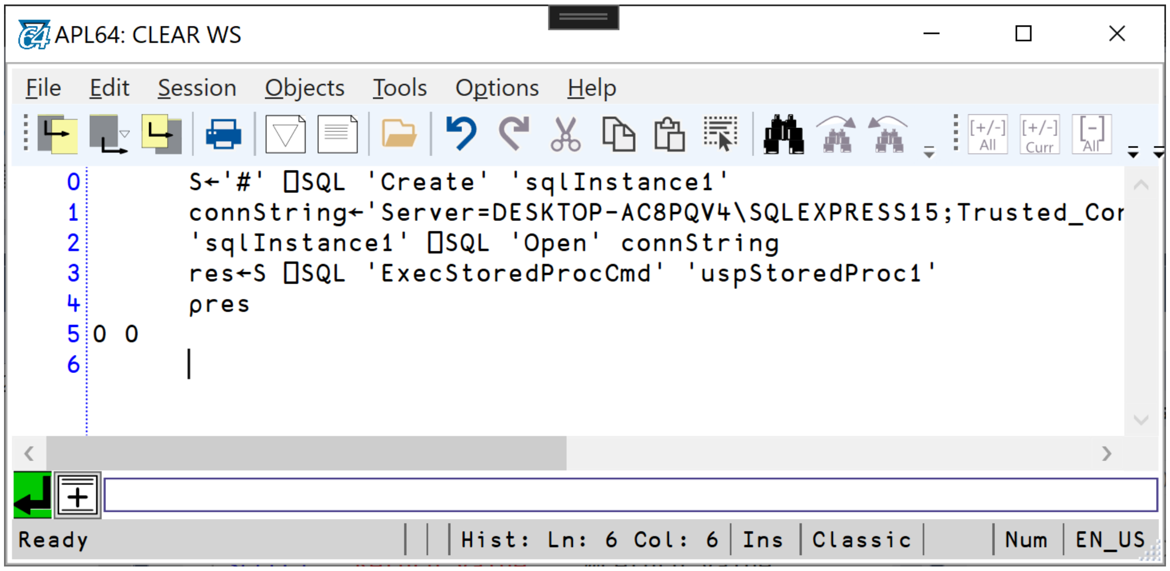
```
CREATE PROCEDURE [dbo].[spGetCompensationFromName]
-- Add the parameters for the stored procedure here
        @NameMatch nvarchar(45),
        @COMP float OUTPUT
AS
```

```
BEGIN
       -- SET NOCOUNT ON added to prevent extra result sets from
       -- interfering with SELECT statements.
       SET NOCOUNT ON;

    -- Insert statements for procedure here
       SELECT @COMP = Compensation from tbOne WHERE Name = @NameMatch
       Select @COMP
END
```

ExecStoredProcCmd

This action can be used to run a stored procedure which has no input or output parameters. The single action argument is an APL text vector (cmdText) containing the stored procedure name. The result is an empty, 2-column APL matrix.

```
res←⎕SQL 'ExecStoredProcCmd' 'uspStoredProc1'
```



```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Cor
      'sqlInstance1' ⎕SQL 'Open' connString
      res←S ⎕SQL 'ExecStoredProcCmd' 'uspStoredProc1'
      ρres
0 0
```

GetAllRecords

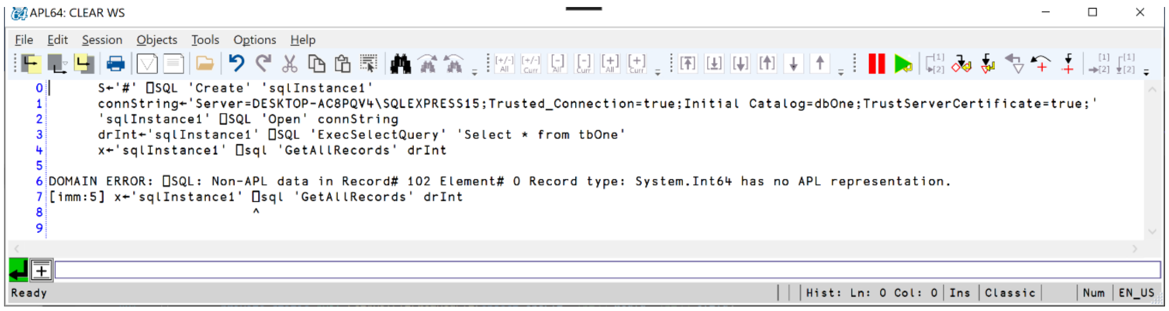
This action applies to an ⎕SQL instance. The 'GetAllRecords' action will get the remaining records which have not already been read from the specified record set. This action closes the specified record set, because all records have been read.

An exception may occur if the data in the record set has no representation in APL64. In this case, the query which created the record set must be modified, e.g. Cast(), to make the returned data conformable with APL64.

An exception will occur if there are no remaining records to be read.

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
```

```
'sqlInstance1' ⎕SQL 'Open' connString
drInt←'sqlInstance1' ⎕SQL 'ExecSelectQuery' 'Select * from tbOne'
x←'sqlInstance1' ⎕sql 'GetAllRecords' drInt
```

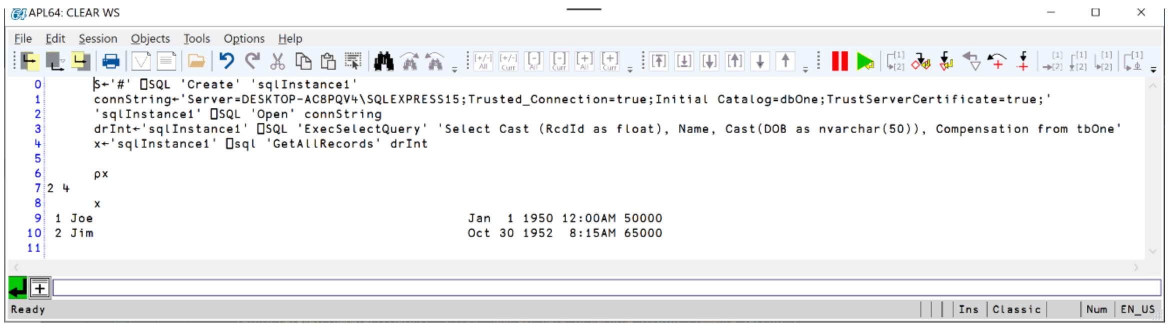


```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      drInt←'sqlInstance1' ⎕SQL 'ExecSelectQuery' 'Select * from tbOne'
      x←'sqlInstance1' ⎕sql 'GetAllRecords' drInt

DOMAIN ERROR: ⎕SQL: Non-APL data in Record# 102 Element# 0 Record type: System.Int64 has no APL representation.
[imm:5] x←'sqlInstance1' ⎕sql 'GetAllRecords' drInt
                          ^
```

In the sample database this particular query failed because the data types being returned in the record set have no representation in APL. Recall that APL provides double, string, integer and bool data types, whereas Microsoft SQL Server includes many additional data types.

To be effective, the SQL Select statement will need to be more sophisticated.

```
    S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
drInt←'sqlInstance1' ⎕SQL 'ExecSelectQuery' 'Select Cast (RcdId as float), Name, Cast(DOB
as nvarchar(50)), Compensation from tbOne'
x←'sqlInstance1' ⎕sql 'GetAllRecords' drInt
```



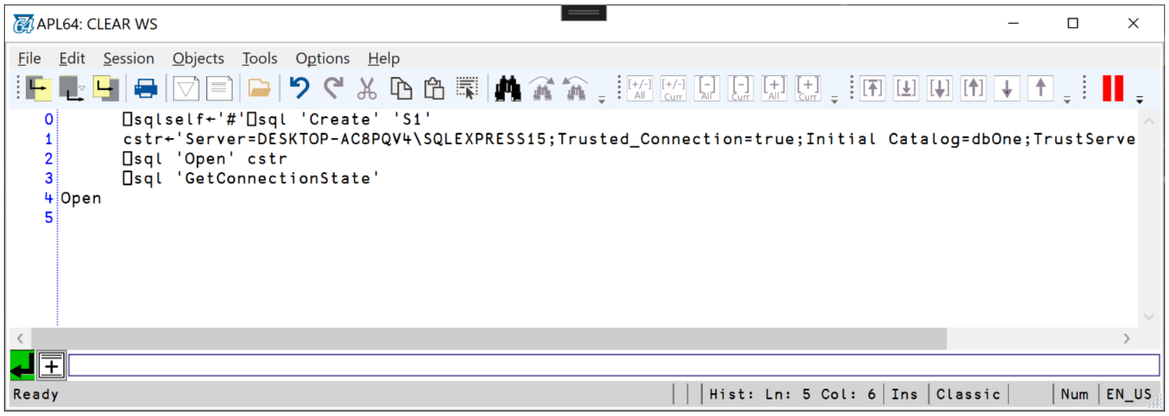
```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      drInt←'sqlInstance1' ⎕SQL 'ExecSelectQuery' 'Select Cast (RcdId as float), Name, Cast(DOB as nvarchar(50)), Compensation from tbOne'
      x←'sqlInstance1' ⎕sql 'GetAllRecords' drInt

      ⍴x
2 4
      x
 1 Joe                                        Jan  1 1950 12:00AM 50000
 2 Jim                                        Oct 30 1952  8:15AM 65000
```

GetConnectionState

This action applies to an ⎕SQL instance. The result of the GetConnectionState action is a text vector indicating the connection state of the specified ⎕SQL instance to an Sql database.

```
⎕sqlself←'#'⎕sql 'Create' 'S1'
cstr←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'
```

```
⎕sql 'Open' cstr
⎕sql 'GetConnectionState'
```

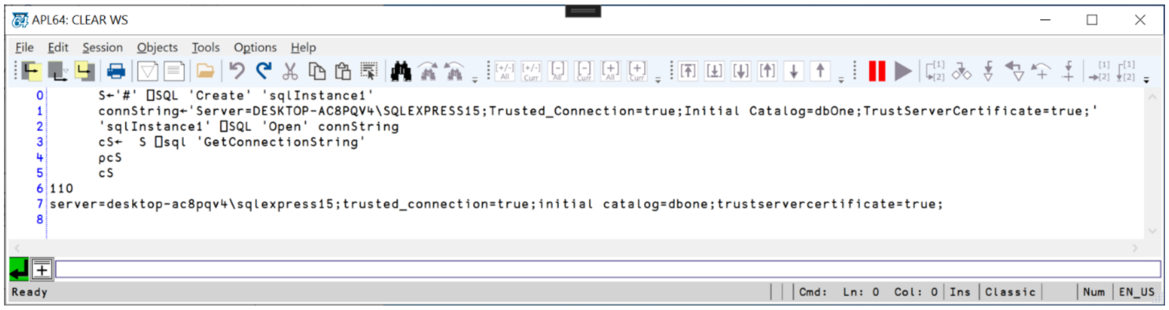


```
      ⎕sqlself←'#'⎕sql 'Create' 'S1'
      cstr←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServe
      ⎕sql 'Open' cstr
      ⎕sql 'GetConnectionState'
Open
```

GetConnectionString

This action applies to an ⎕SQL instance. The result of the GetConnectionString is the Sql connection string, if any, associated with the specified ⎕SQL instance. Use the Open action to set the connection string.

```
    S←'#' ⎕SQL 'Create' 'sqlInstance1'
    connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

   'sqlInstance1' ⎕SQL 'Open' connString
    cS← S ⎕sql 'GetConnectionString'
    ⍴cS
    cS
```



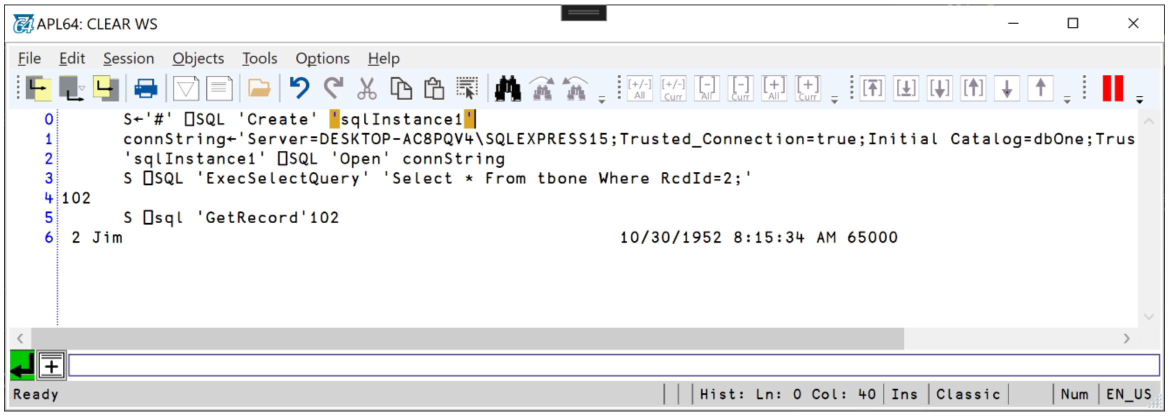
```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'sqlInstance1' ⎕SQL 'Open' connString
      cS←  S ⎕sql 'GetConnectionString'
      ⍴cS
      cS
110
server=desktop-ac8pqv4\sqlexpress15;trusted_connection=true;initial catalog=dbone;trustservercertificate=true;
```

GetRecord

This action applies to an ⎕SQL instance. The right argument is the ⎕SQL record set number. The GetRecord action will read the next available record, if any, from the specified record set and return its values to the APL64 instance.

An exception may occur if the data in the record set has no representation in APL64. In this case the query which created the record set must be modified to make the returned data conformable with APL64. If such an exception does occur, the Cast() method of the database can generally convert the source data in the database record to an APL64 data type.

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

 'sqlInstance1' ⎕SQL 'Open' connString
S ⎕SQL 'ExecSelectQuery' 'Select * From tbone Where RcdId=2;'
S ⎕sql 'GetRecord'102
```

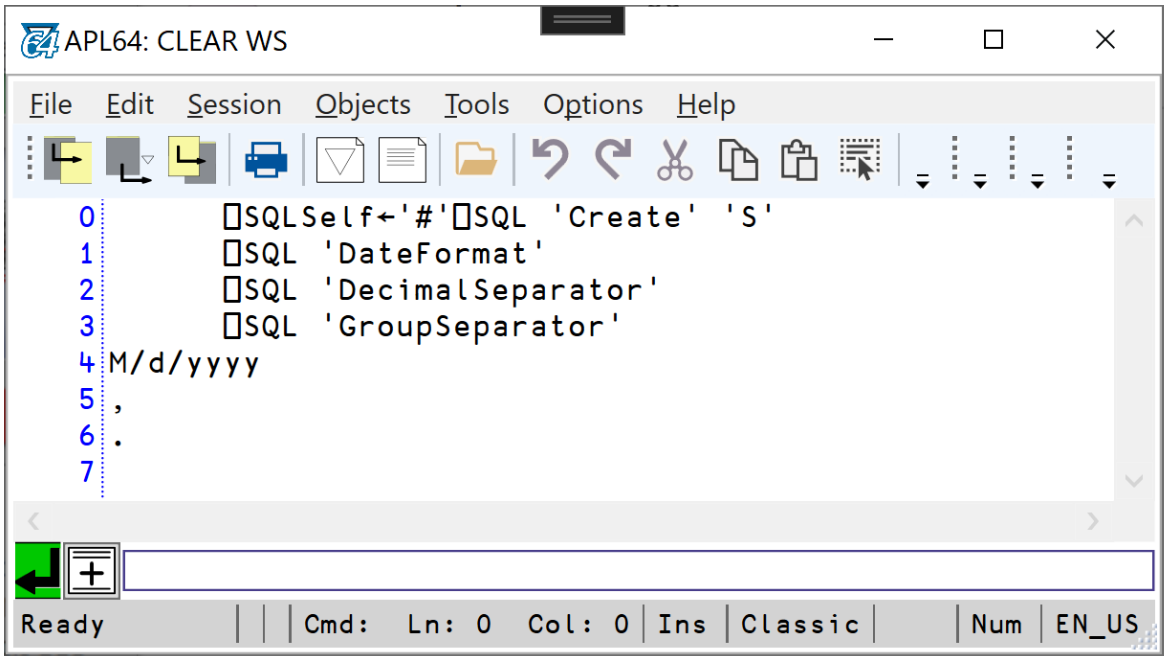


```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;Trus
      'sqlInstance1' ⎕SQL 'Open' connString
      S ⎕SQL 'ExecSelectQuery' 'Select * From tbone Where RcdId=2;'
102
      S ⎕sql 'GetRecord'102
 2 Jim                                                   10/30/1952 8:15:34 AM 65000
```

GroupSeparator

This action returns the .Net Number Group Separator text applicable to the specified ⎕SQL instance.

```
⎕SQLSelf←'#'⎕SQL 'Create' 'S'
⎕SQL 'DateFormat'
⎕SQL 'DecimalSeparator'
⎕SQL 'GroupSeparator'
```


APL64: CLEAR WS
File Edit Session Objects Tools Options Help

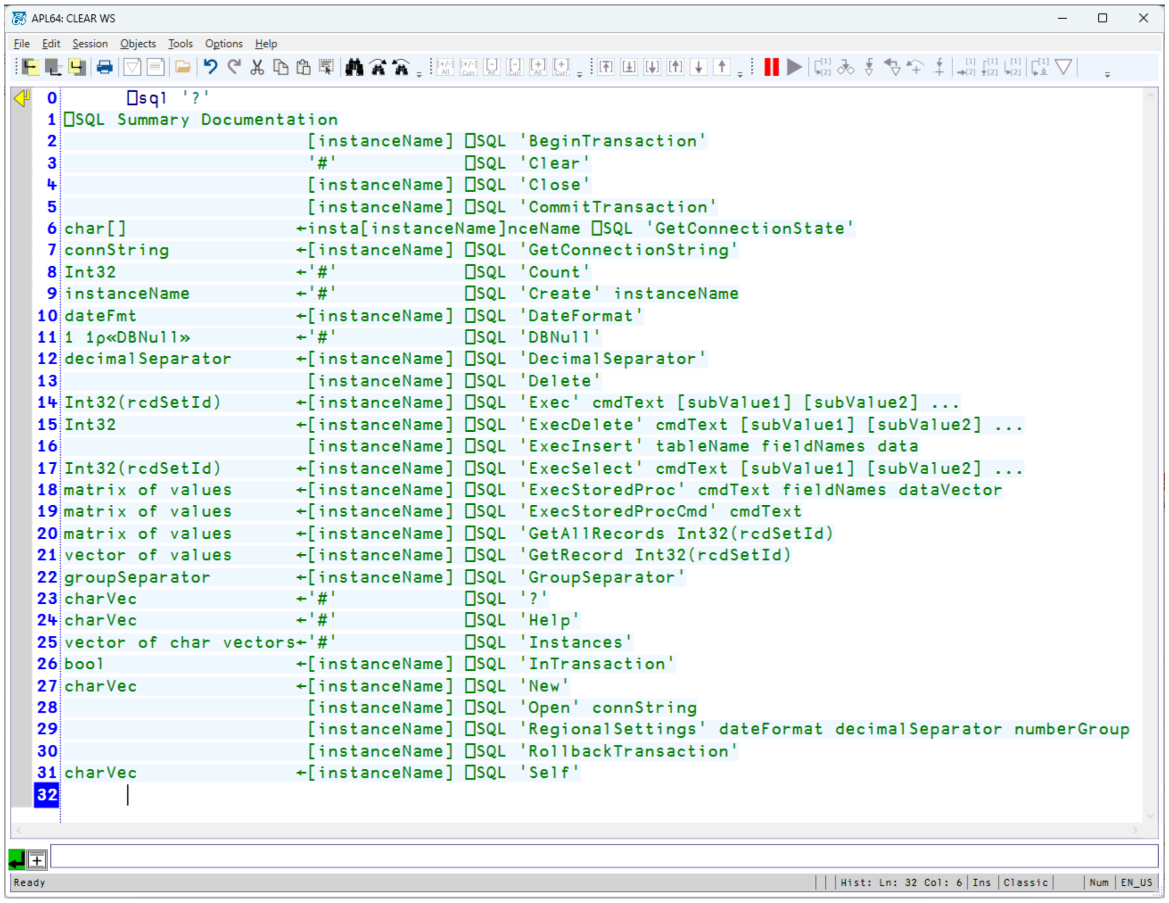
```
      ⎕SQLSelf←'#'⎕SQL 'Create' 'S'
      ⎕SQL 'DateFormat'
      ⎕SQL 'DecimalSeparator'
      ⎕SQL 'GroupSeparator'
M/d/yyyy
,
.
```

Ready Cmd: Ln: 0 Col: 0 Ins Classic Num EN_US

? or Help

This action is performed on the ⎕SQL object. The ? and Help actions will return a text array containing a summary of the ⎕SQL syntax.

```
⎕SQL '?'
```

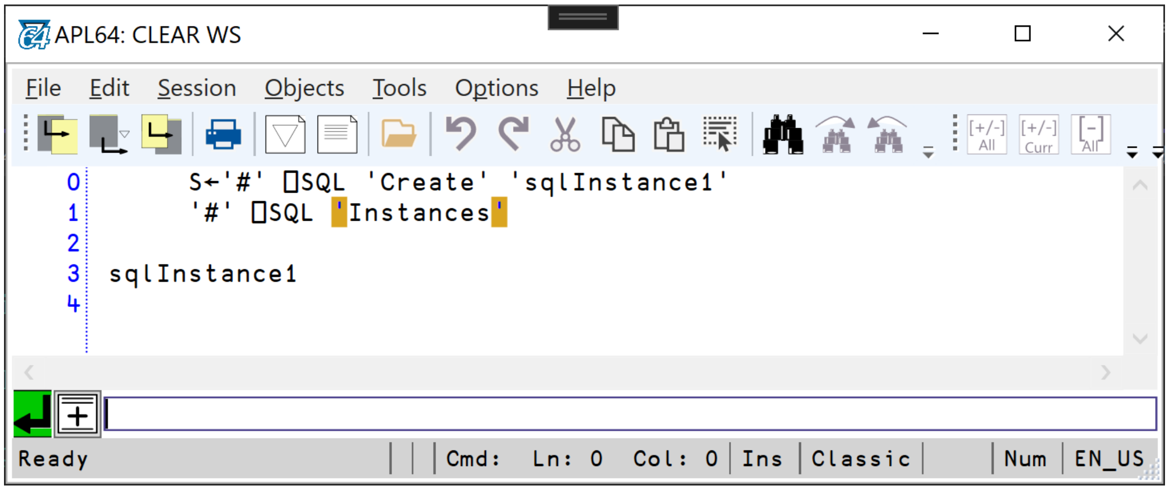


```
      ⎕sql '?'
⎕SQL Summary Documentation
                       [instanceName] ⎕SQL 'BeginTransaction'
                       '#'            ⎕SQL 'Clear'
                       [instanceName] ⎕SQL 'Close'
                       [instanceName] ⎕SQL 'CommitTransaction'
char[]                ←insta[instanceName]nceName ⎕SQL 'GetConnectionState'
connString            ←[instanceName] ⎕SQL 'GetConnectionString'
Int32                 ←'#'            ⎕SQL 'Count'
instanceName          ←'#'            ⎕SQL 'Create' instanceName
dateFmt               ←[instanceName] ⎕SQL 'DateFormat'
1 1ρ«DBNull»          ←'#'            ⎕SQL 'DBNull'
decimalSeparator      ←[instanceName] ⎕SQL 'DecimalSeparator'
                       [instanceName] ⎕SQL 'Delete'
Int32(rcdSetId)       ←[instanceName] ⎕SQL 'Exec' cmdText [subValue1] [subValue2] ...
Int32                 ←[instanceName] ⎕SQL 'ExecDelete' cmdText [subValue1] [subValue2] ...
                       [instanceName] ⎕SQL 'ExecInsert' tableName fieldNames data
Int32(rcdSetId)       ←[instanceName] ⎕SQL 'ExecSelect' cmdText [subValue1] [subValue2] ...
matrix of values      ←[instanceName] ⎕SQL 'ExecStoredProc' cmdText fieldNames dataVector
matrix of values      ←[instanceName] ⎕SQL 'ExecStoredProcCmd' cmdText
matrix of values      ←[instanceName] ⎕SQL 'GetAllRecords Int32(rcdSetId)
vector of values      ←[instanceName] ⎕SQL 'GetRecord Int32(rcdSetId)
groupSeparator        ←[instanceName] ⎕SQL 'GroupSeparator'
charVec               ←'#'            ⎕SQL '?'
charVec               ←'#'            ⎕SQL 'Help'
vector of char vectors←'#'            ⎕SQL 'Instances'
bool                  ←[instanceName] ⎕SQL 'InTransaction'
charVec               ←[instanceName] ⎕SQL 'New'
                       [instanceName] ⎕SQL 'Open' connString
                       [instanceName] ⎕SQL 'RegionalSettings' dateFormat decimalSeparator numberGroup
                       [instanceName] ⎕SQL 'RollbackTransaction'
charVec               ←[instanceName] ⎕SQL 'Self'
```

Instances

This action is performed on the ⎕SQL object. The result is a vector of char vectors representing a list of instance names.

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
'#' ⎕SQL 'Instances'
```

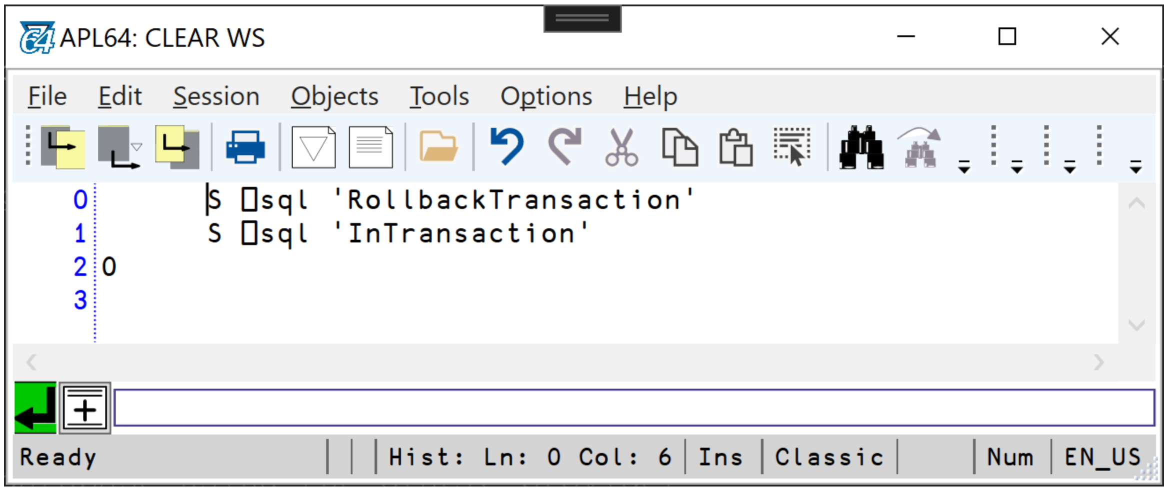


```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      '#' ⎕SQL 'Instances'

 sqlInstance1
```

InTransaction

This action applies to an ⎕SQL instance. The Boolean result of the InTransaction action indicates if a transaction block is in existence for the specified ⎕SQL instance.

```
⎕SQL 'RollbackTransaction'
⎕SQL 'InTransaction'
```



APL64: CLEAR WS
File Edit Session Objects Tools Options Help

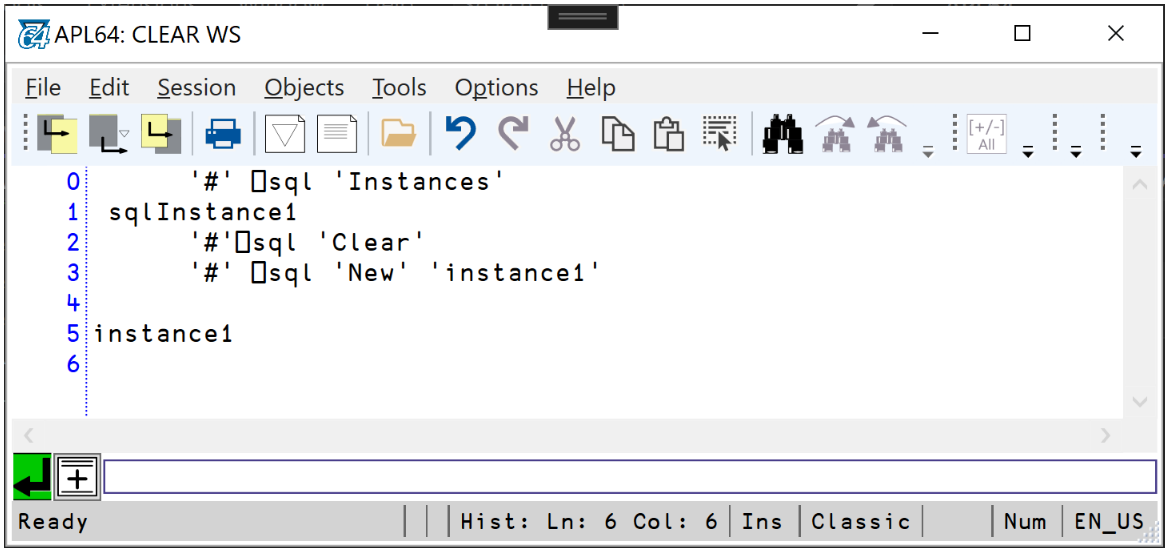
```
      ⎕sql 'RollbackTransaction'
      ⎕sql 'InTransaction'
0
```

Ready | Hist: Ln: 0 Col: 6 | Ins | Classic | Num | EN_US

New

This action is performed on the ⎕SQL object. The New action will create an ⎕SQL instance with a user-provided name in the right argument. Multiple ⎕SQL instances are possible in the same APL64 instance, so that multiple Sql databases may be conveniently accessed. The New action will fail if the named instance already exists. The New action for a particular Sql database is generally used once in an APL64 instance. The New action does not open a connection to an Sql database, use the Open action for that purpose. The result of a successful 'New' action is a text vector containing the ⎕SQL instance name.

```
'#'⎕SQL 'Clear'
'#'⎕SQL 'New' 'instance1'
```

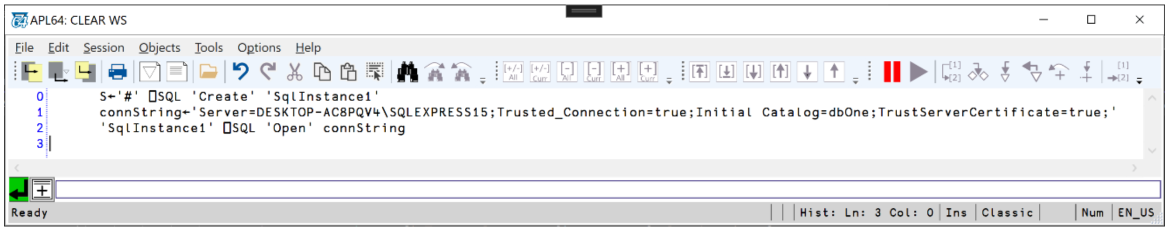


```
      '#' ⎕sql 'Instances'
sqlInstance1
      '#'⎕sql 'Clear'
      '#' ⎕sql 'New' 'instance1'

instance1
```

Open

This action applies to an ⎕SQL instance. The Open action requires the specification of a connection string as the right argument.

```
S←'#' ⎕SQL 'Create' 'SqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'SqlInstance1' ⎕SQL 'Open' connString
```



```
      S←'#' ⎕SQL 'Create' 'SqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      'SqlInstance1' ⎕SQL 'Open' connString
```

RegionalSettings

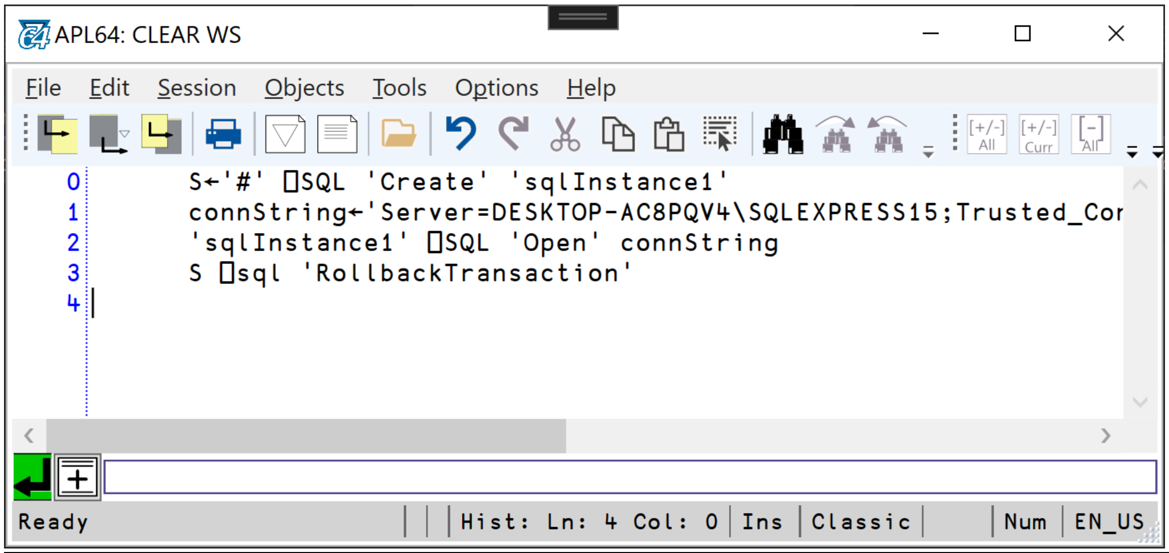
- This action can be used to set the regional values for the specified ⎕SQL instance:
- ShortDatePattern
- CurrencyDecimalSeparator, NumberDecimalSeparator & PercentDecimalSeparator
- CurrencyGroupSeparator, NumberGroupSeparator & PercentGroupSeparator

```
'instance1' ⎕sql 'RegionalSettings' 'M/d/yyyy' '.' ','
```

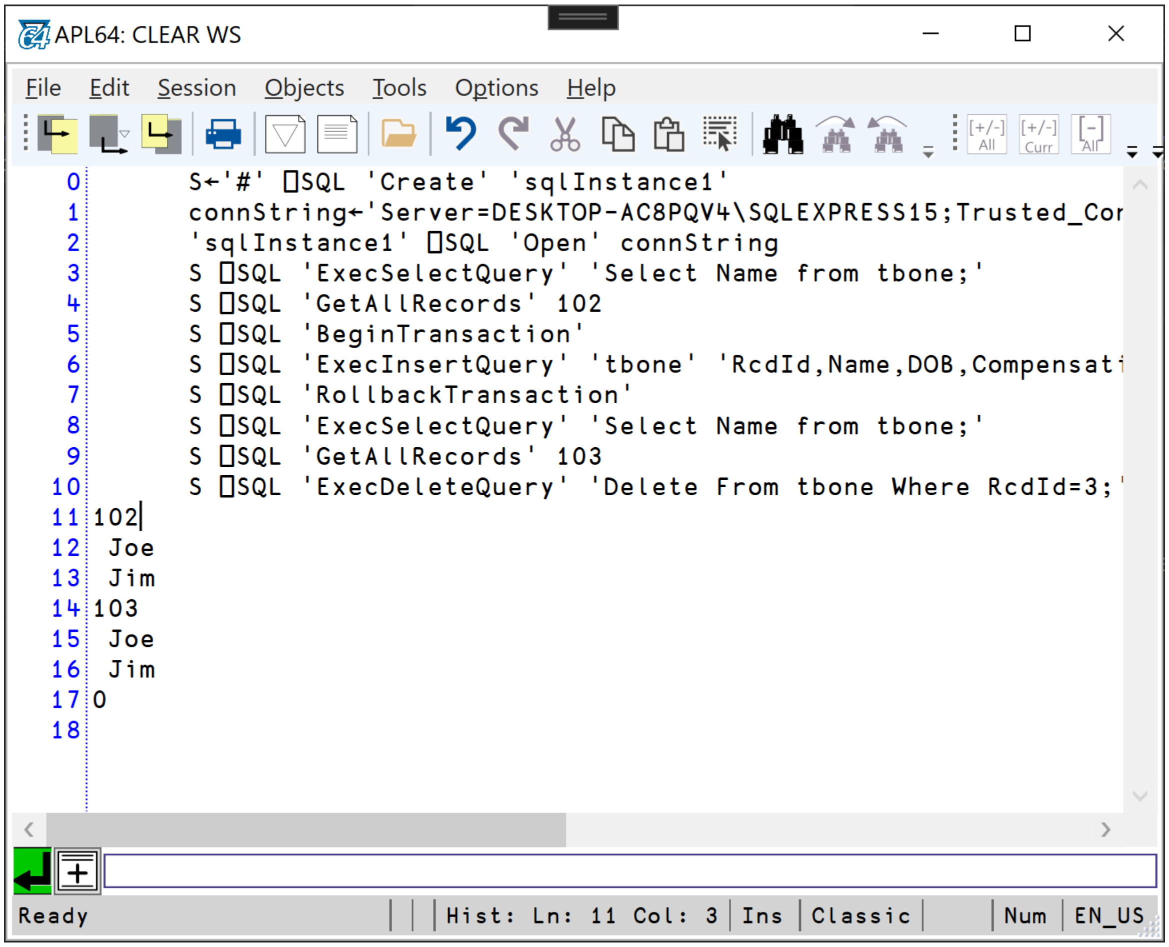
RollbackTransaction

This action applies to an ⎕SQL instance. The RollbackTransaction action is used to cancel any pending SQL operations which are included in the current SQL transaction. The RollbackTransaction action has no result. The RollbackTransaction action has no effect if no SQL transaction is in progress.

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
S ⎕sql 'RollbackTransaction'
```


```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Cor
'sqlInstance1' ⎕SQL 'Open' connString
S ⎕sql 'RollbackTransaction'
```

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connStringS ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
S ⎕SQL 'GetAllRecords' 102
S ⎕SQL 'BeginTransaction'
S ⎕SQL 'ExecInsertQuery' 'tbone' 'RcdId,Name,DOB,Compensation' (3 'Salmon' '1966-01-01'
90500.56)
S ⎕SQL 'RollbackTransaction'
S ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
S ⎕SQL 'GetAllRecords' 103
S ⎕SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
```



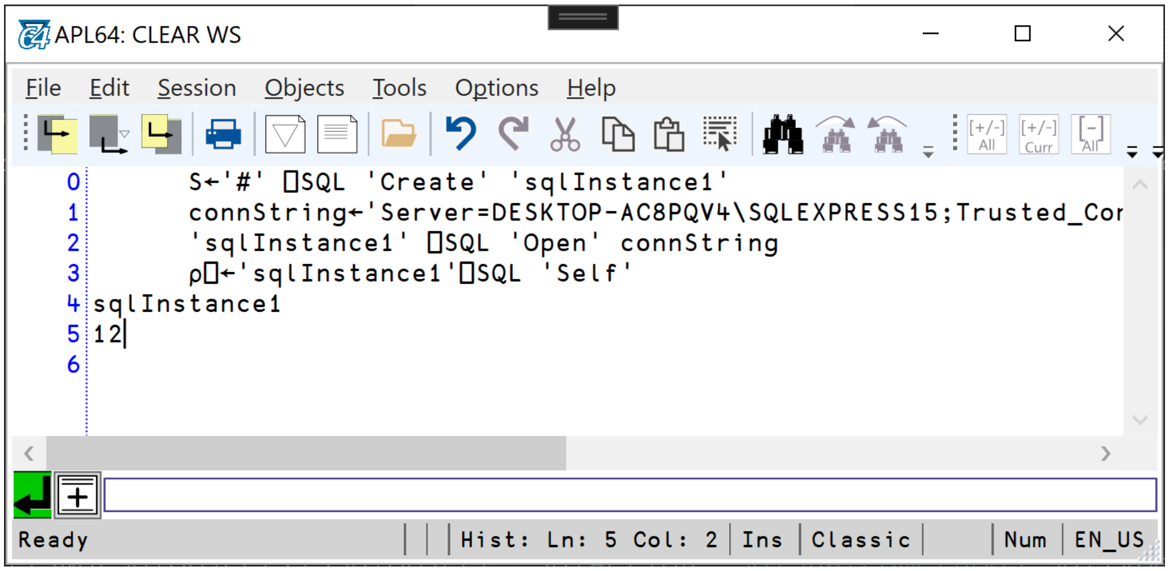
```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Cor
      'sqlInstance1' ⎕SQL 'Open' connString
      S ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
      S ⎕SQL 'GetAllRecords' 102
      S ⎕SQL 'BeginTransaction'
      S ⎕SQL 'ExecInsertQuery' 'tbone' 'RcdId,Name,DOB,Compensati
      S ⎕SQL 'RollbackTransaction'
      S ⎕SQL 'ExecSelectQuery' 'Select Name from tbone;'
      S ⎕SQL 'GetAllRecords' 103
      S ⎕SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
102
 Joe
 Jim
103
 Joe
 Jim
0
```

```
Ready    Hist: Ln: 11 Col: 3  Ins  Classic    Num  EN_US
```

Self

This action is performed on the ⎕SQL object. The right argument specifies an ⎕SQL instance name for a potentially existing ⎕SQL instance. If the ⎕SQL instance exists in the APL64 instance, the result of the Self action is the ⎕SQL instance name.

```
S←'#' ⎕SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ⎕SQL 'Open' connString
ρ⎕←'sqlInstance1'⎕SQL 'Self'
```



```
      S←'#' ⎕SQL 'Create' 'sqlInstance1'
      connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Cor
      'sqlInstance1' ⎕SQL 'Open' connString
      ρ⎕←'sqlInstance1'⎕SQL 'Self'
sqlInstance1
12
```

Learn Structured Query Language

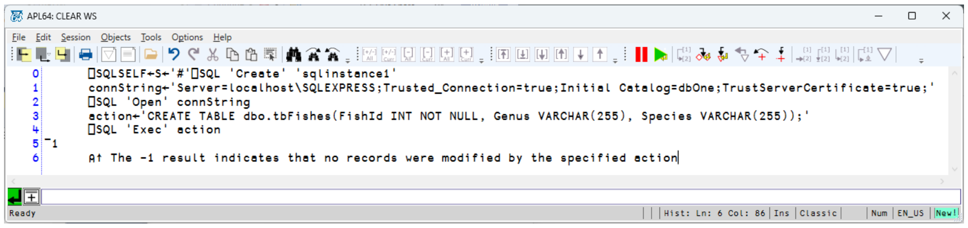
http://www.w3schools.com/sql/default.asp

Some Additional Examples

Create Table

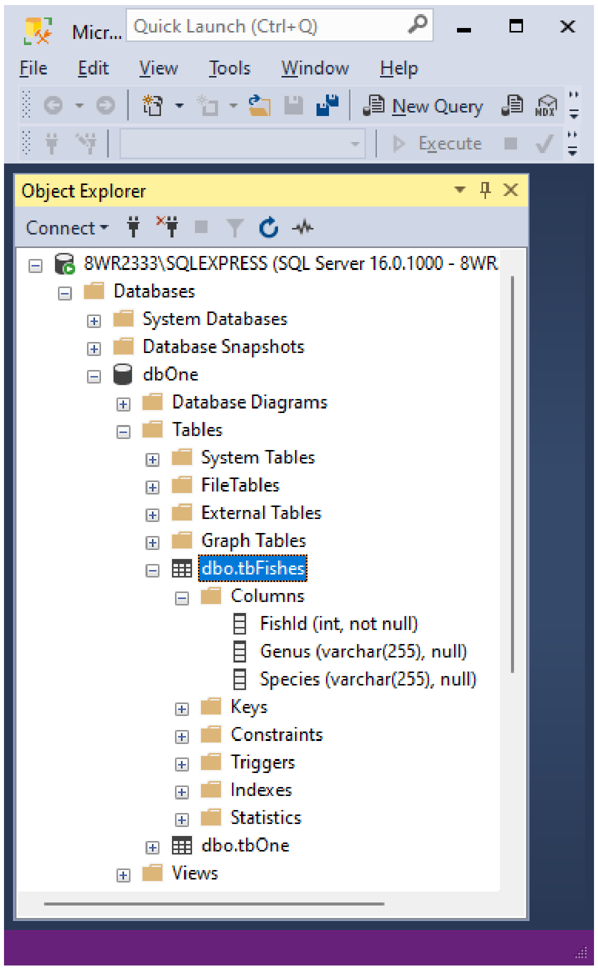
Use the ⎕SQL 'Exec' action to create a new table within an existing database using an appropriate SQL statement.

```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

⎕SQL 'Open' connString
action←'CREATE TABLE dbo.tbFishes(FishId INT NOT NULL, Genus VARCHAR(255), Species
VARCHAR(255));'
⎕SQL 'Exec' action
```



```
      ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
      connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      ⎕SQL 'Open' connString
      action←'CREATE TABLE dbo.tbFishes(FishId INT NOT NULL, Genus VARCHAR(255), Species VARCHAR(255));'
      ⎕SQL 'Exec' action
¯1

      A← The -1 result indicates that no records were modified by the specified action
```

Verify that the table was created using Microsoft SQL Server Management Studio:

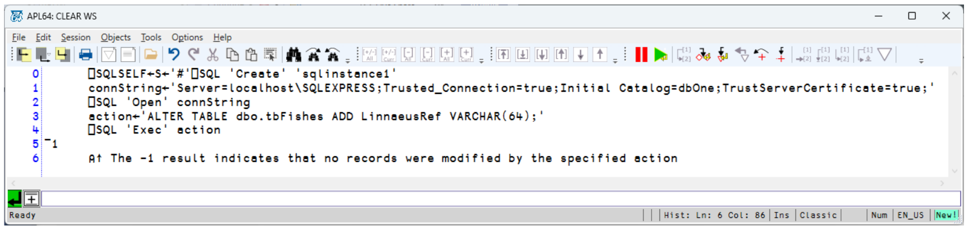


```
8WR2333\SQLEXPRESS (SQL Server 16.0.1000 - 8WR
  Databases
    System Databases
    Database Snapshots
    dbOne
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.tbFishes
          Columns
            FishId (int, not null)
            Genus (varchar(255), null)
            Species (varchar(255), null)
          Keys
          Constraints
          Triggers
          Indexes
          Statistics
        dbo.tbOne
      Views
```

Alter Table

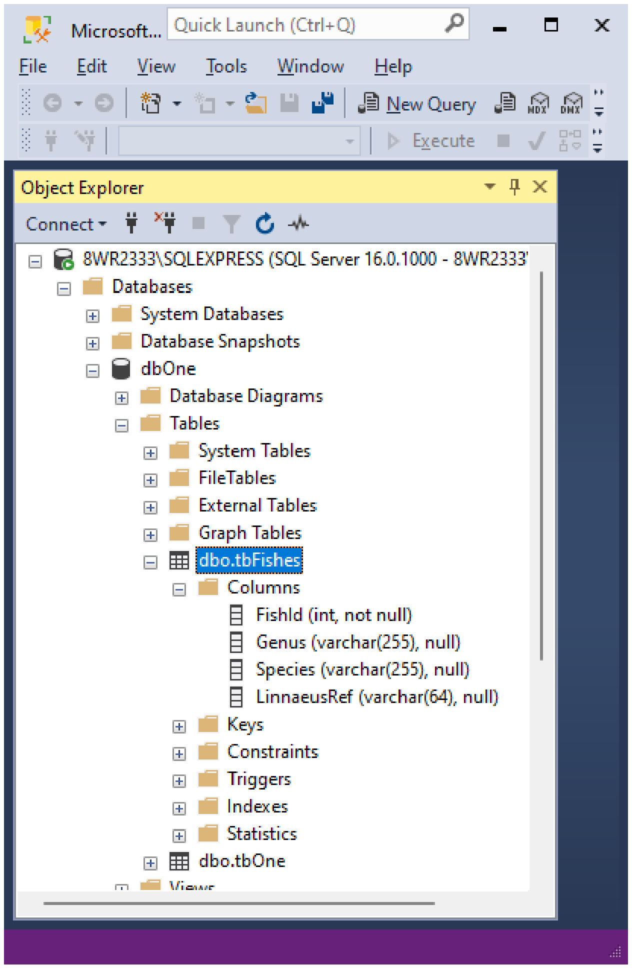
Use the ⎕SQL 'Exec' action using an Alter Table SQL statement to modify an existing table in an existing database. In this example, a new column is added to an existing table using an appropriate SQL statement.

```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

⎕SQL 'Open' connString
action←'ALTER TABLE dbo.tbFishes ADD LinnaeusRef VARCHAR(64);'
⎕SQL 'Exec' action
```



```
      ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
      connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      ⎕SQL 'Open' connString
      action←'ALTER TABLE dbo.tbFishes ADD LinnaeusRef VARCHAR(64);'
      ⎕SQL 'Exec' action
¯1
      ⍝ The -1 result indicates that no records were modified by the specified action
```

Verify that the column was added using Microsoft SQL Server Management Studio:



```
8WR2333\SQLEXPRESS (SQL Server 16.0.1000 - 8WR2333'
  Databases
    System Databases
    Database Snapshots
    dbOne
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.tbFishes
          Columns
            FishId (int, not null)
            Genus (varchar(255), null)
            Species (varchar(255), null)
            LinnaeusRef (varchar(64), null)
          Keys
          Constraints
          Triggers
          Indexes
          Statistics
        dbo.tbOne
      Views
```

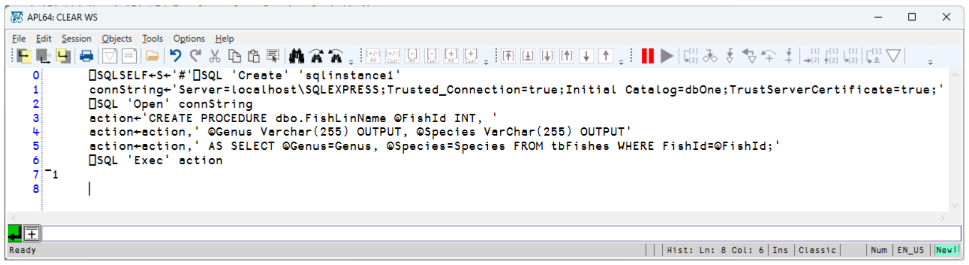
Create/Alter Procedure

This example uses the tbFishes table created in the previous Create Table and Alster Table examples.

Use the ⎕SQL 'Exec' action using a Create Procedure or Alter Procedure statement to create or alter a stored procedure. In this example a new stored procedure is added to an existing database using an appropriate SQL statement:

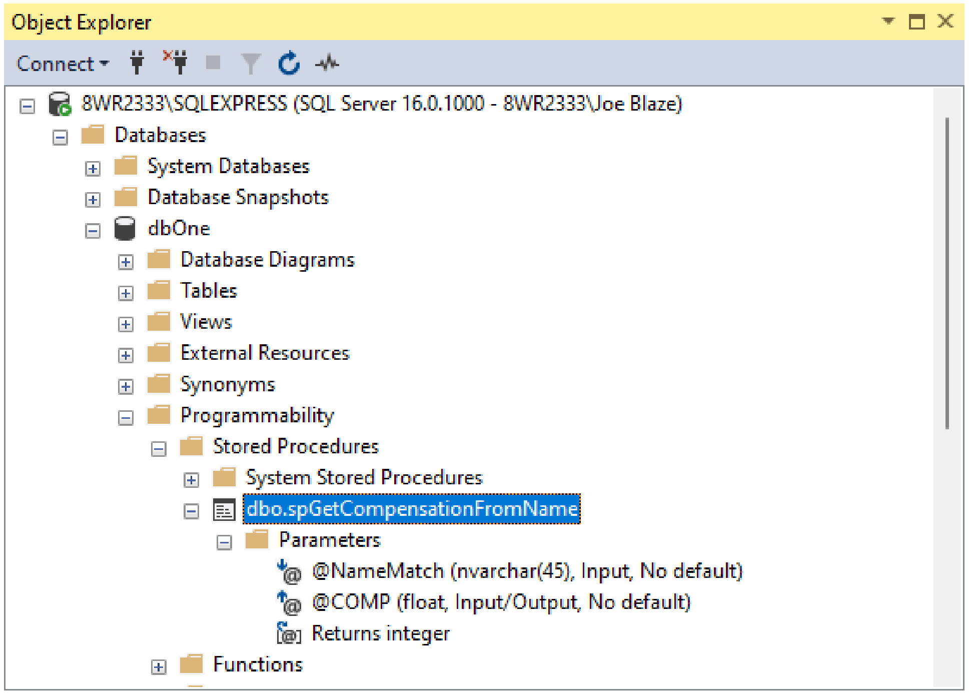
```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
```

```
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

⎕SQL 'Open' connString
action←'CREATE PROCEDURE dbo.FishLinName @FishId INT, '
action←action,' @Genus Varchar(255) OUTPUT, @Species VarChar(255) OUTPUT'
action←action,' AS SELECT @Genus=Genus, @Species=Species FROM tbFishes WHERE
FishId=@FishId;'
⎕SQL 'Exec' action
```


APL64: CLEAR WS
File Edit Session Objects Tools Options Help
Ready
Hist: Ln: 8 Col: 6 Ins Classic Num EN_US New!

```
      ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
      connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      ⎕SQL 'Open' connString
      action←'CREATE PROCEDURE dbo.FishLinName @FishId INT, '
      action←action,' @Genus Varchar(255) OUTPUT, @Species VarChar(255) OUTPUT'
      action←action,' AS SELECT @Genus=Genus, @Species=Species FROM tbFishes WHERE FishId=@FishId;'
      ⎕SQL 'Exec' action
¯1
```

Verify the stored procedure is created using Microsoft SQL Server Management Studio:

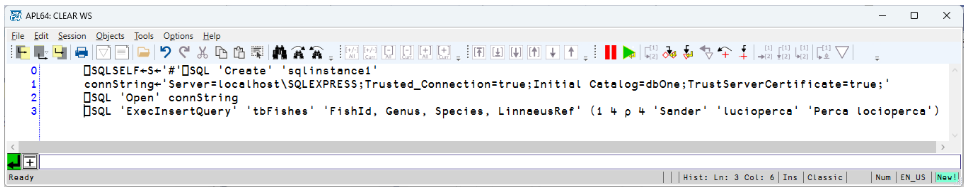


```
8WR2333\SQLEXPRESS (SQL Server 16.0.1000 - 8WR2333\Joe Blaze)
  Databases
    System Databases
    Database Snapshots
    dbOne
      Database Diagrams
      Tables
      Views
      External Resources
      Synonyms
      Programmability
        Stored Procedures
          System Stored Procedures
          dbo.spGetCompensationFromName
            Parameters
              @NameMatch (nvarchar(45), Input, No default)
              @COMP (float, Input/Output, No default)
              Returns integer
        Functions
```

Add a data record to the tbFishes table:

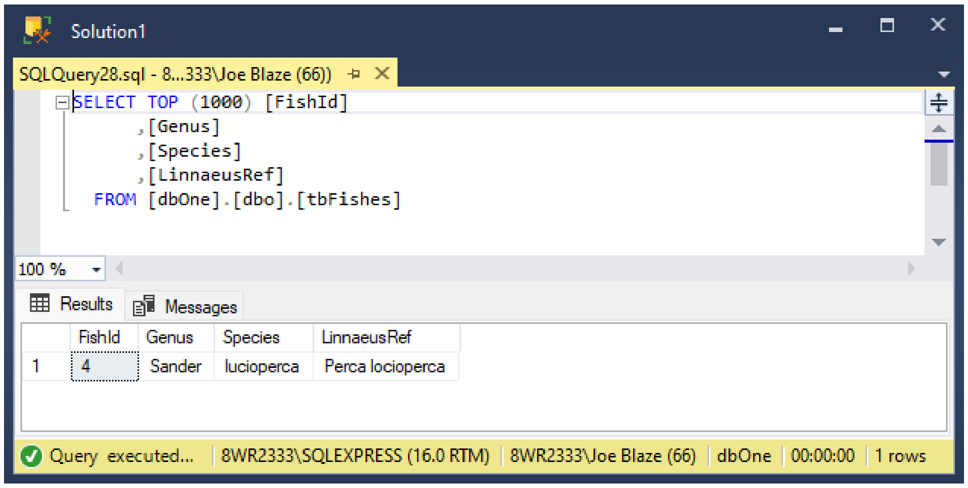
```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

⎕SQL 'Open' connString
```

```
⎕SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 ρ 4 'Sander'
'lucioperca' 'Perca locioperca')
```



```
        ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
        connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
        ⎕SQL 'Open' connString
        ⎕SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 ρ 4 'Sander' 'lucioperca' 'Perca locioperca')
```

Verify the data record was inserted using Microsoft SQL Server Management Studio:



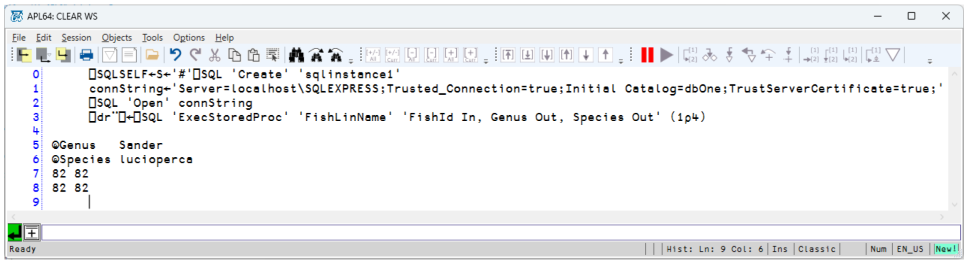
```
SELECT TOP (1000) [FishId]
      ,[Genus]
      ,[Species]
      ,[LinnaeusRef]
  FROM [dbOne].[dbo].[tbFishes]
```

	FishId	Genus	Species	LinnaeusRef
1	4	Sander	lucioperca	Perca locioperca

Query executed... | 8WR2333\SQLEXPRESS (16.0 RTM) | 8WR2333\Joe Blaze (66) | dbOne | 00:00:00 | 1 rows

Run the stored procedure in APL64:

```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
 connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

 ⎕SQL 'Open' connString
 ⎕dr¨⎕←⎕SQL 'ExecStoredProc' 'FishLinName' 'FishId In, Genus Out, Species Out' (1ρ4)
```



```
        ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
        connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
        ⎕SQL 'Open' connString
        ⎕dr¨⎕←⎕SQL 'ExecStoredProc' 'FishLinName' 'FishId In, Genus Out, Species Out' (1ρ4)

 @Genus   Sander
 @Species lucioperca
 82 82
 82 82
```

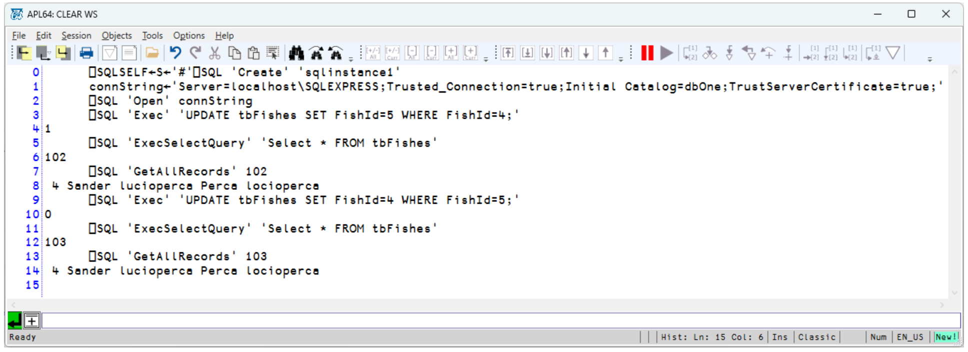
Update Table Record

This example uses the tbFishes table created in the previous Create Table, Alter Table and Create/Alter Procedure examples.

Method #1:

Use the ⎕SQL ‘Exec’ action with an appropriate SQL UPDATE statement:

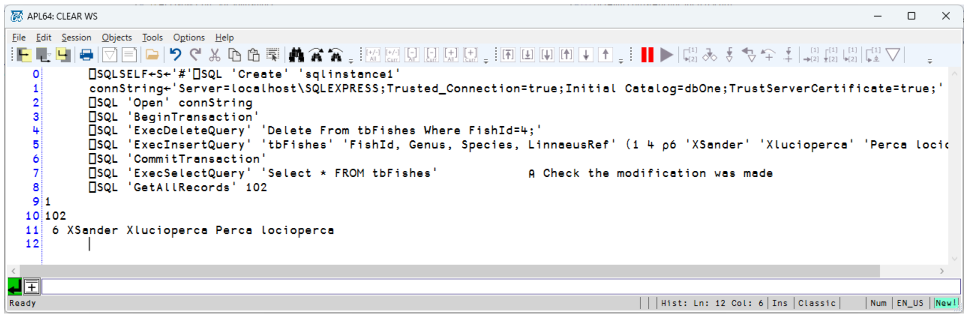
```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
 connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

 ⎕SQL 'Open' connString
 ⎕SQL 'Exec' 'UPDATE tbFishes SET FishId=5 WHERE FishId=4;' ⍝ Modify a column value for a
specific record
 ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'        ⍝ Check the modification was made
 ⎕SQL 'GetAllRecords' 102
 ⎕SQL 'Exec' 'UPDATE tbFishes SET FishId=4 WHERE FishId=5;' ⍝ Reverse the modification
 ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'        ⍝ Check the reversal was made
 ⎕SQL 'GetAllRecords' 103
```



```
      ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
      connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      ⎕SQL 'Open' connString
      ⎕SQL 'Exec' 'UPDATE tbFishes SET FishId=5 WHERE FishId=4;'
1
      ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'
102
      ⎕SQL 'GetAllRecords' 102
 4 Sander lucioperca Perca locioperca
      ⎕SQL 'Exec' 'UPDATE tbFishes SET FishId=4 WHERE FishId=5;'
0
      ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'
103
      ⎕SQL 'GetAllRecords' 103
 4 Sander lucioperca Perca locioperca
```

Method #2:
Within an SQL transaction, delete the desired table record and insert the modified table record

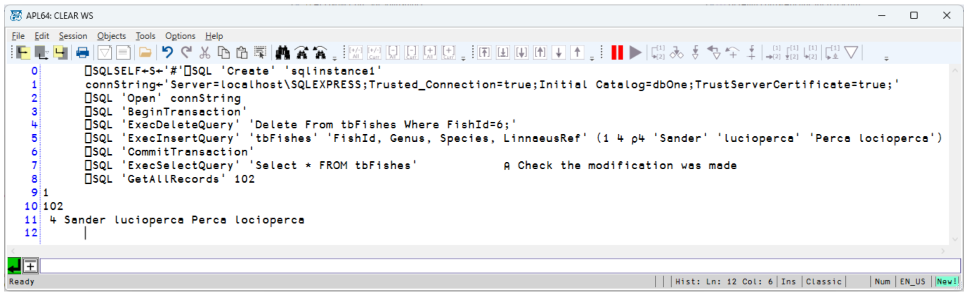
```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
 connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

 ⎕SQL 'Open' connString
 ⎕SQL 'BeginTransaction'
 ⎕SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=4;'
 ⎕SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 ρ6 'XSander'
'Xlucioperca' 'Perca locioperca')
 ⎕SQL 'CommitTransaction'
 ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'        ⍝ Check the modification was made
 ⎕SQL 'GetAllRecords' 102
```



```
      ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
      connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      ⎕SQL 'Open' connString
      ⎕SQL 'BeginTransaction'
      ⎕SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=4;'
      ⎕SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 ρ6 'XSander' 'Xlucioperca' 'Perca locic
      ⎕SQL 'CommitTransaction'
      ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'           ⍝ Check the modification was made
      ⎕SQL 'GetAllRecords' 102
1
102
 6 XSander Xlucioperca Perca locioperca
```

After completing this example, restore the tbFishes record state:

```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
 connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

 ⎕SQL 'Open' connString
 ⎕SQL 'BeginTransaction'
 ⎕SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=6;'
 ⎕SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 ρ4 'Sander'
'lucioperca' 'Perca locioperca')
 ⎕SQL 'CommitTransaction'
 ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'          ⍝ Check the modification was made
 ⎕SQL 'GetAllRecords' 102
```



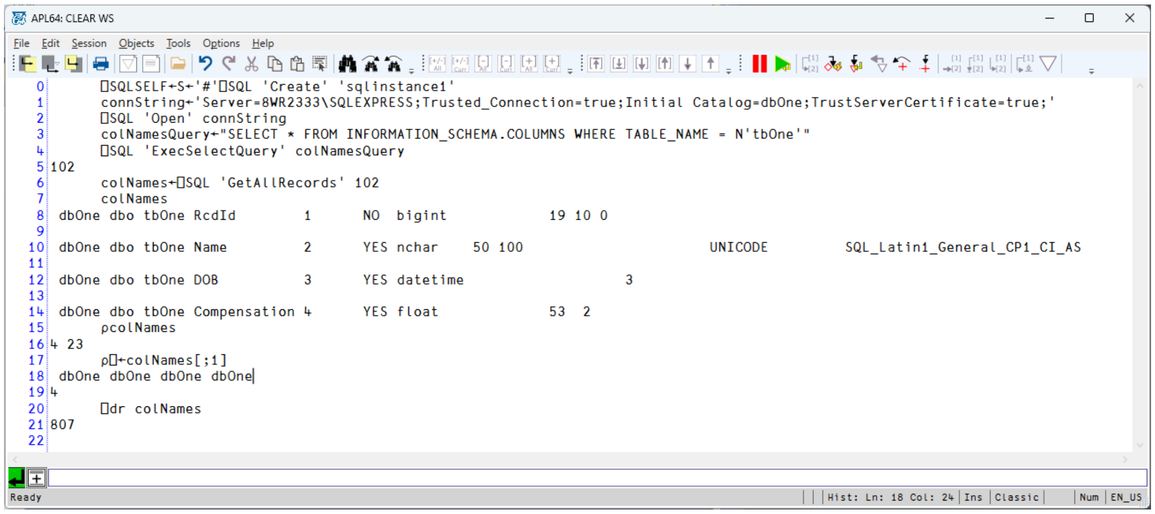
```
      ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
      connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      ⎕SQL 'Open' connString
      ⎕SQL 'BeginTransaction'
      ⎕SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=6;'
      ⎕SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 ρ4 'Sander' 'lucioperca' 'Perca locioperca')
      ⎕SQL 'CommitTransaction'
      ⎕SQL 'ExecSelectQuery' 'Select * FROM tbFishes'           ⍝ Check the modification was made
      ⎕SQL 'GetAllRecords' 102
1
102
 4 Sander lucioperca Perca locioperca
```

Obtain Columns Information for a Table

SQL queries can be made to ‘built-in’ tables in the database, e.g. obtain column information for a programmer-created table:

```
⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
 connString←'Server=8WR2333\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

 ⎕SQL 'Open' connString
colNamesQuery←"SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME
= N'tbOne'"

⎕SQL 'ExecSelectQuery' colNamesQuery
```

```
colNames←⎕SQL 'GetAllRecords' 102
colNames
ρcolNames
ρ⎕←colNames[;1]
⎕dr colNames
```



APL64: CLEAR WS

```
      ⎕SQLSELF←S←'#'⎕SQL 'Create' 'sqlinstance1'
      connString←'Server=8WR2333\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
      ⎕SQL 'Open' connString
      colNamesQuery←"SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = N'tbOne'"
      ⎕SQL 'ExecSelectQuery' colNamesQuery
102
      colNames←⎕SQL 'GetAllRecords' 102
      colNames
 dbOne dbo tbOne RcdId        1      NO  bigint            19 10 0

 dbOne dbo tbOne Name         2      YES nchar    50 100                       UNICODE        SQL_Latin1_General_CP1_CI_AS

 dbOne dbo tbOne DOB          3      YES datetime                    3

 dbOne dbo tbOne Compensation 4      YES float             53  2
      ρcolNames
4 23
      ρ⎕←colNames[;1]
 dbOne dbOne dbOne dbOne
4
      ⎕dr colNames
807
```

Ready | Hist: Ln: 18 Col: 24 | Ins | Classic | Num | EN_US