# Using SQLITE

## Contents

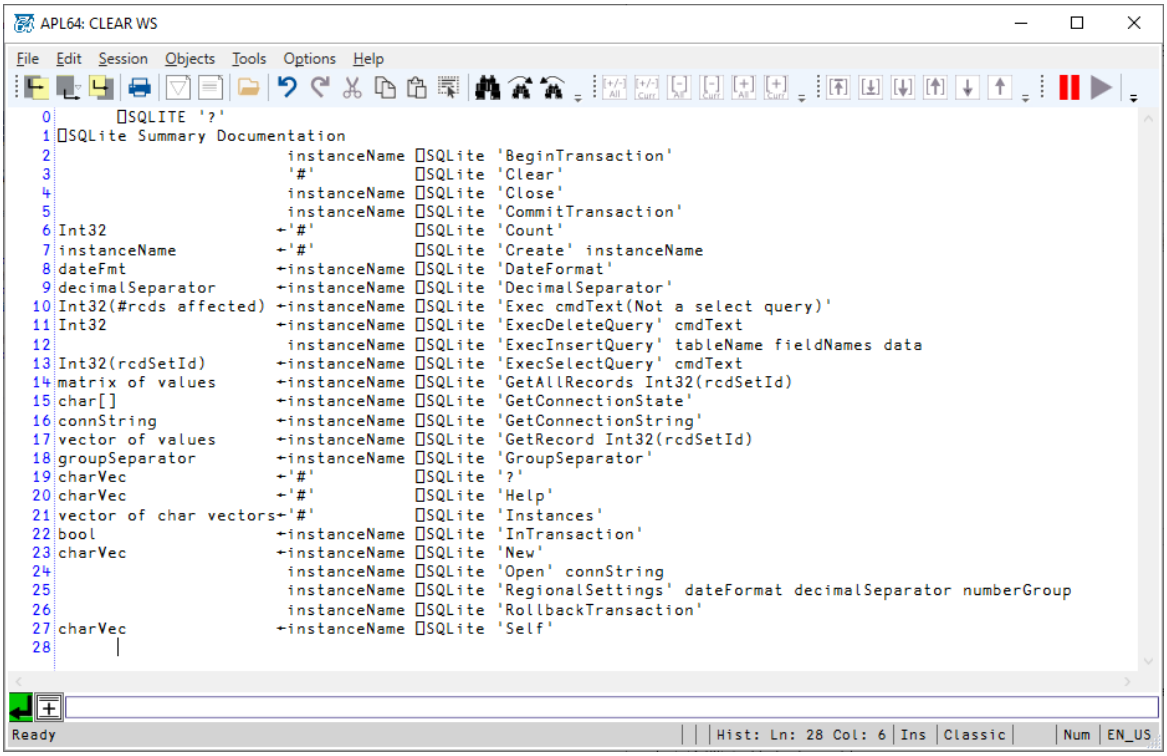
## Overview

The APL64 □SQLITE system function provides a structured query language (SQL)-based interface to an Sqlite database.

The Tool employs the System.Data.SQLite .Net assembly so APL64 executable statements may be used to Execute, Select, Delete and Insert components of an SQLITE database.

With the (no-cost) availability of the Sqlite software, scalable-to-enterprise-level database software is more affordable and a choice for professionally-designed application systems.

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




## Prerequisites

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

The examples in this document assume that a sample SQLITE database, named 'TEST.DB' is installed, running and accessible to the target workstation. The examples in this document also assume that this sample SQLITE 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 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 SQLITE

Sqlite is an open source product, so the fully-functional Sqlite product may be installed by anyone complying with its license agreement.

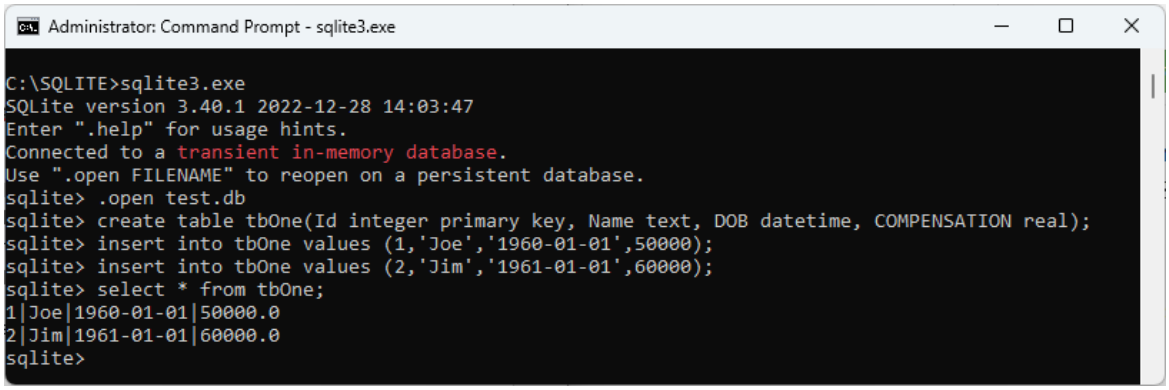
Full installation details are beyond the scope of this document. The SQLITE getting started documentation and the SQLITE download page can be helpful

- Copy the pre-compiled SQLITE binaries to a folder on the target workstation
- Copy the command line tools bundle, including the sqlite3.exe program, to the same folder

After installing SQLITE, for purposes of the examples in this document, use the sqlite3.exe program to:

- Create the c:\sqlite\test.db database
- Create the tbOne table
- Insert sample records
- Verify the sample data

```
Sqlite3.exe
.open test.db
create table tbOne(Id integer primary key, Name text, DOB datetime, COMPENSATION real);
insert into tbOne values (1,'Joe','1960-01-01',50000);
insert into tbOne values (2,'Jim','1961-01-01',60000);
select * from tbOne;
```




## SQLITE Data Types

Recent versions of Sqlite support two data type methodologies, strict and dynamic.

If the strict option is not used when a table is created, the data type of a field in a record is determined by the data type of the field value rather than the data type of the table column.

If the strict option is used when a table is created the available data types are restricted to INTEGER, TEXT, BLOB, REAL and NUMERIC.

The □SQLITE system function uses the Sqlite data type affinity rules to determine the data type to be used when converting APL-based data to be used when inserting an Sqlite record.

APL64 has no representation of some Sqlite data types, which should be considered when specifying Sqlite table.

## □SQLITE Actions

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

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

The name of an □sqlite instance can be specified by:

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

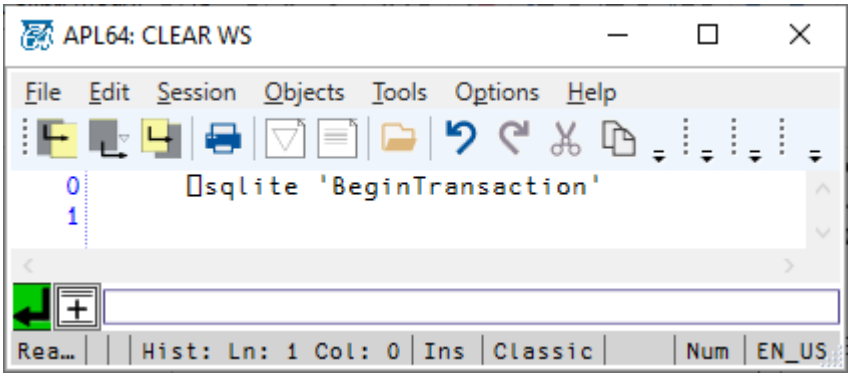
□sqlite action names are not case sensitive. □sqlite instance names are case sensitive.

## BeginTransaction

This action applies to an □sqlite instance. The BeginTransaction action, for the Sqlite instance named in the left argument, indicates that subsequent □sqlite actions will be incorporated into an SQL transaction. An SQL transaction may contain multiple □sqlite 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.

```
□sqlite 'BeginTransaction'
```

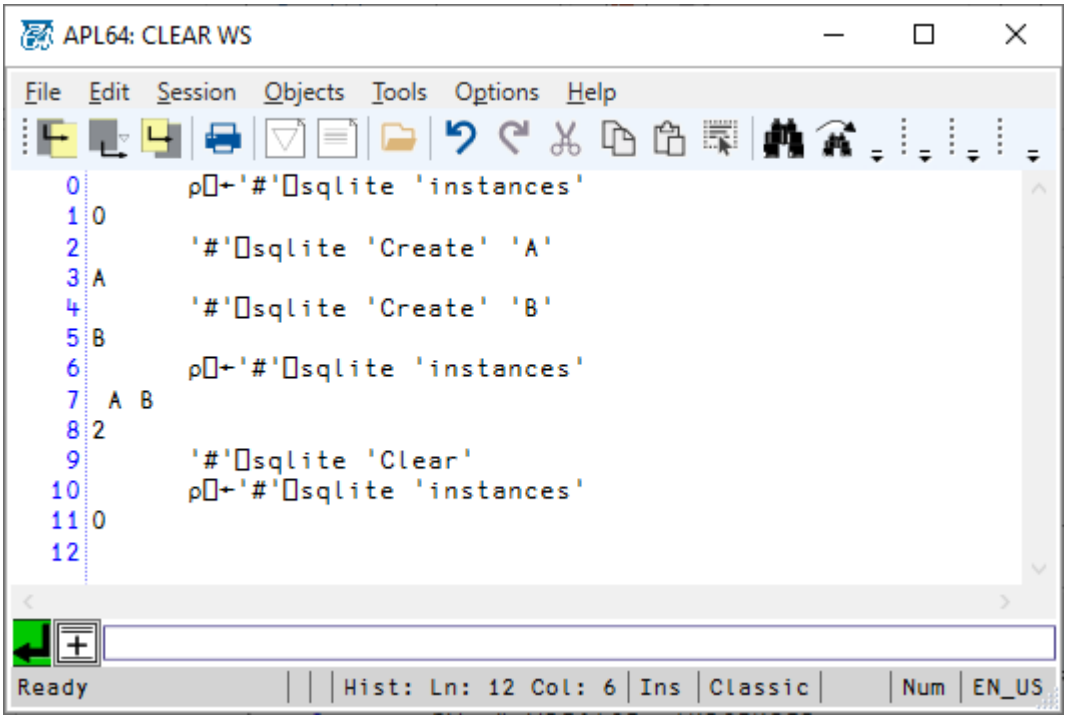



## Clear

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

```
ρ□←'#'□sqlite 'instances'
 '#'□sqlite 'Create' 'A'
 '#'□sqlite 'Create' 'B'
ρ□←'#'□sqlite 'instances'
'#'□sqlite 'Clear'
```

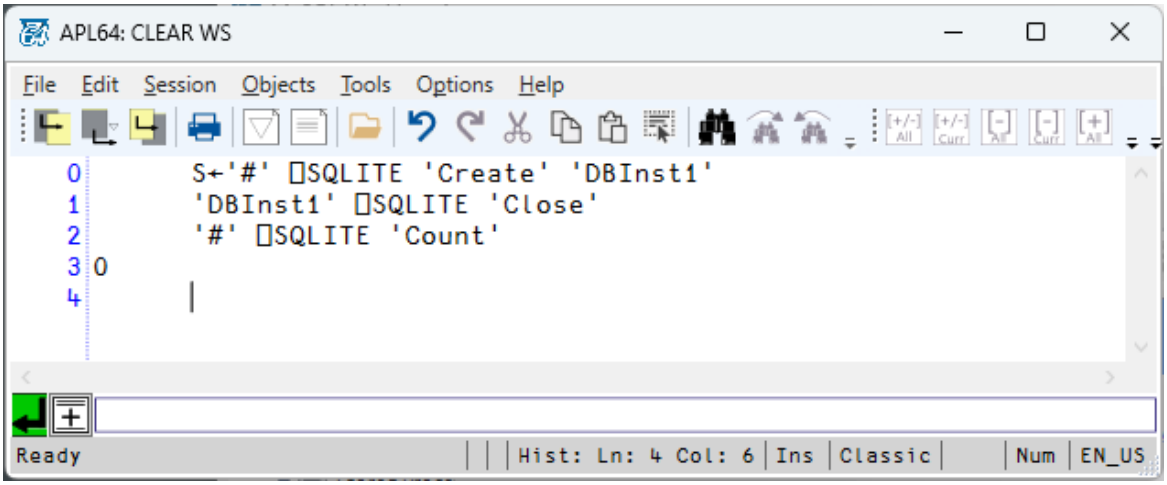
```
ρ⎕←'#'⎕sqlite 'instances'
```




## Close

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

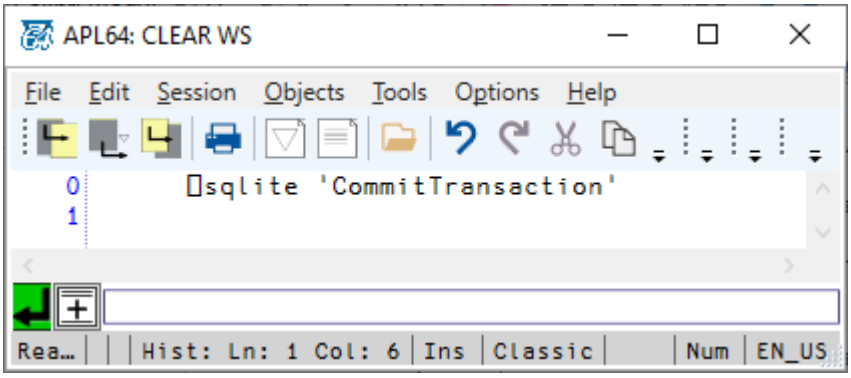
```
S←'#' ⎕SQLITE 'Create' 'DBInst1'
'DBInst1' ⎕SQLITE 'Close'
 '#' ⎕SQLITE 'Count'
```



## CommitTransaction

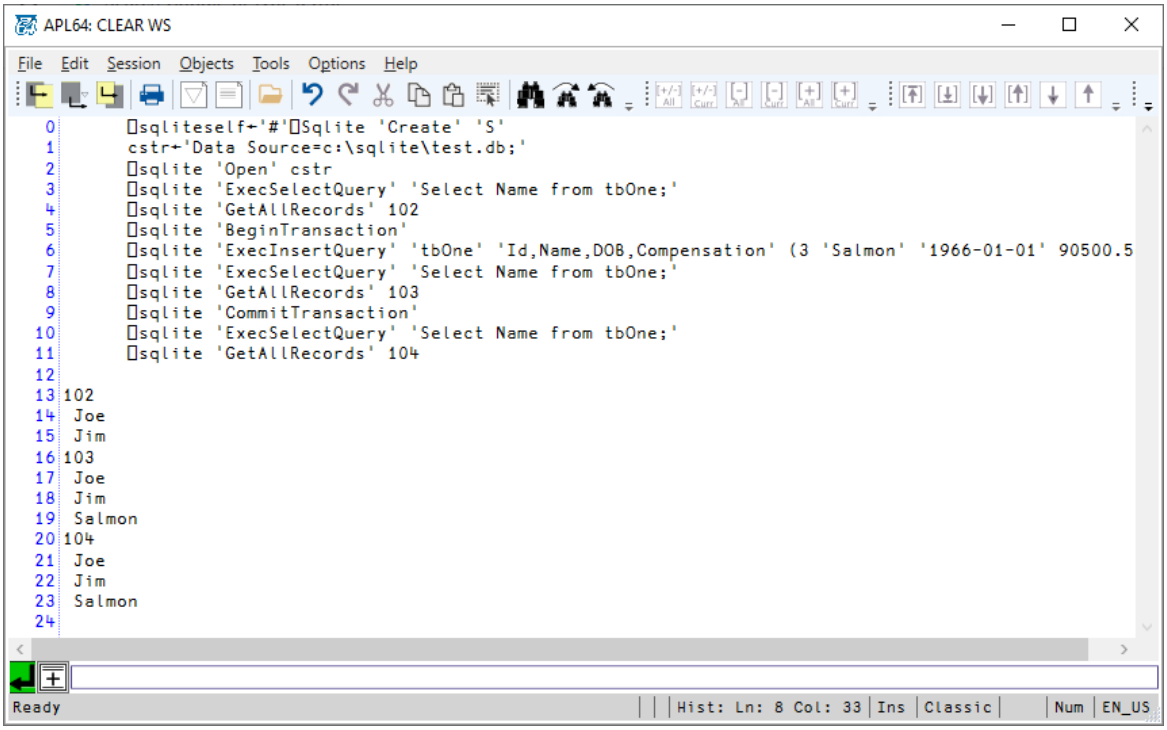
This action applies to an ⎕sqlite instance. The CommitTransaction action causes the effects of the current transaction to be applied to the connected Sqlite database. The CommitTransaction action has no effect if the BeginTransaction action has not occurred or if the RollbackTransaction action has occurred.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

```
⎕sqlite 'CommitTransaction'
```




```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
⎕sqlite 'GetAllRecords' 102
⎕sqlite 'BeginTransaction'
⎕sqlite 'ExecInsertQuery' 'tbOne' 'Id,Name,DOB,Compensation' (3 'Salmon' '1966-01-01' 90500.56)
⎕sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
⎕sqlite 'GetAllRecords' 103
⎕sqlite 'CommitTransaction'
⎕sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
⎕sqlite 'GetAllRecords' 104
```

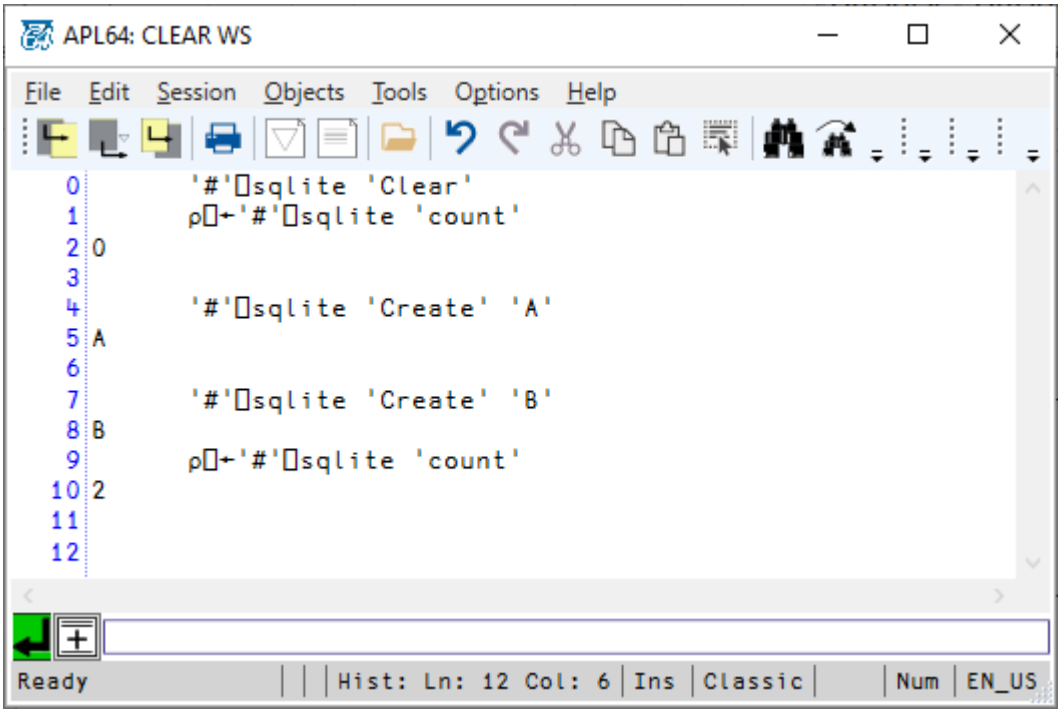
Note that the records in the #103 data set include the pending effects of the ExecInsertQuery action and the records in the #104 data set include the effects of the committed ExecInsertQuery action:




## Count

This action is performed on the ⎕sqlite object. The Count action returns the number of ⎕sqlite instances which exist in the APL64 instance.

```
'#'⎕sqlite 'Clear'
⍴⎕←'#'⎕sqlite 'count'
'#'⎕sqlite 'Create' 'A'
'#'⎕sqlite 'Create' 'B'
⍴⎕←'#'⎕sqlite 'count'
```

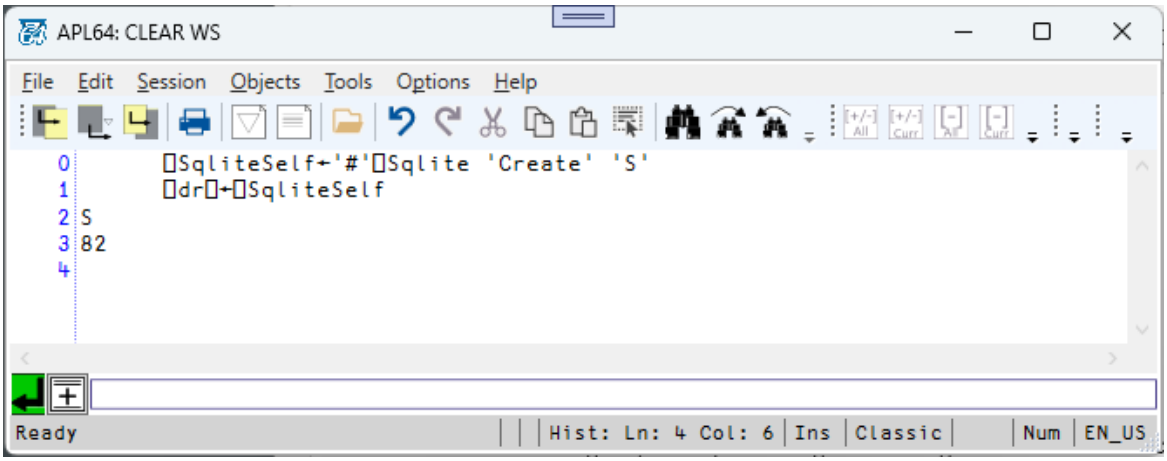



## Create

This action is performed on the ⎕sqlite object. The Create action will create an ⎕Sqlite instance with a user-provided name in the right argument. Multiple ⎕Sqlite instances are possible in the same APL64 instance, so that multiple Sqlite 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 Sqlite database is generally used once in an APL64 instance. The Create action does not open a connection to an Sqlite database, use the Open action for that purpose. The result of a successful 'Create' action is a text vector containing the ⎕Sqlite instance name.

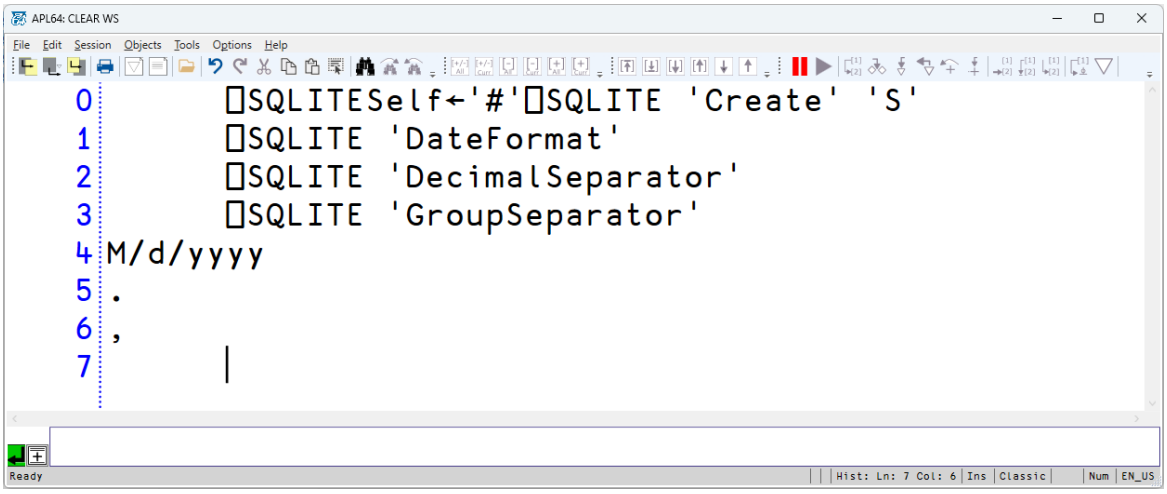
```
⎕SqliteSelf←'#'⎕Sqlite 'Create' 'S'
⎕dr⎕←⎕SqliteSelf
```



## DateFormat

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

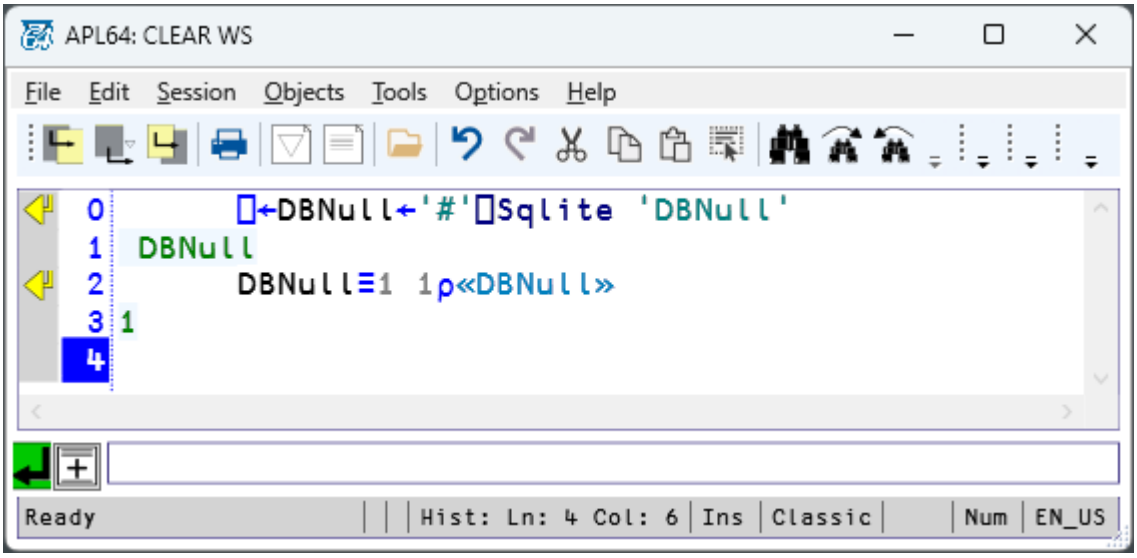
```
⎕SQLITESelf←'#'⎕SQLITE 'Create' 'S'
⎕SQLITE 'DateFormat'
⎕SQLITE 'DecimalSeparator'
⎕SQLITE 'GroupSeparator'
```




## DBNull

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

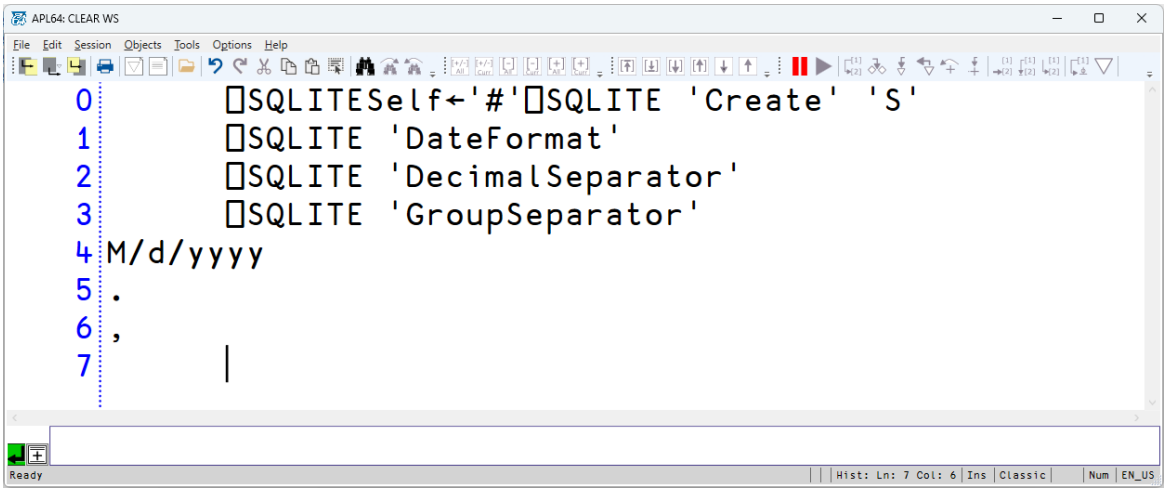
```
⎕←DBNull←'#'⎕Sql 'DBNull'
DBNull≡1 1⍴«DBNull»
```




## DecimalSeparator

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

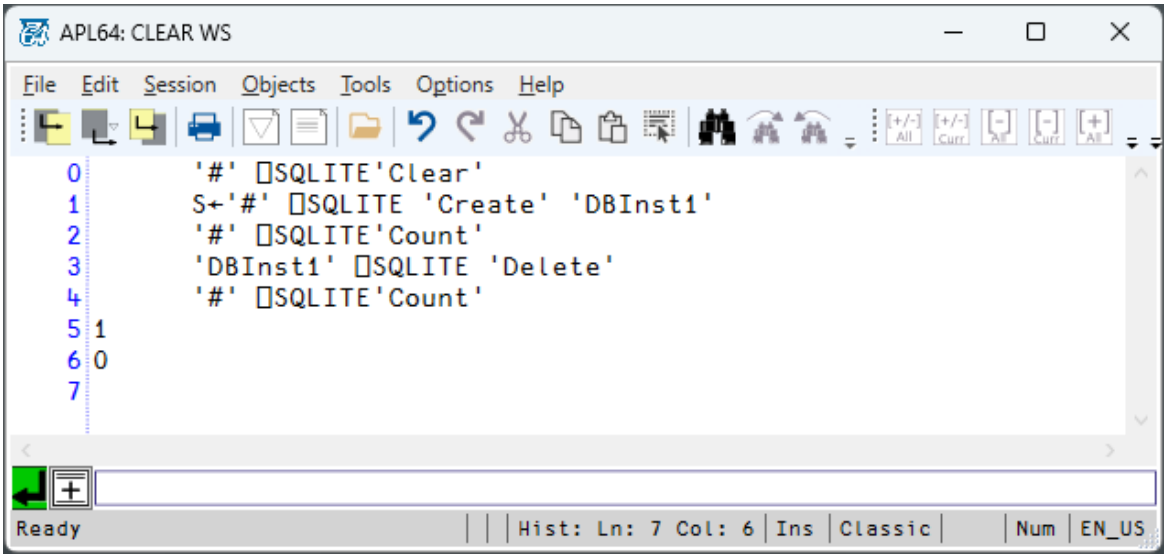
```
⎕SQLITESelf←'#'⎕SQLITE 'Create' 'S'
⎕SQLITE 'DateFormat'
⎕SQLITE 'DecimalSeparator'
⎕SQLITE 'GroupSeparator'
```




## Delete

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

```
'#' ⎕SQLITE'Clear'
S←'#' ⎕SQLITE 'Create' 'DBInst1'
 '#' ⎕SQLITE'Count'
'DBInst1' ⎕SQLITE 'Delete'
 '#' ⎕SQLITE'Count'
```




## Exec

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

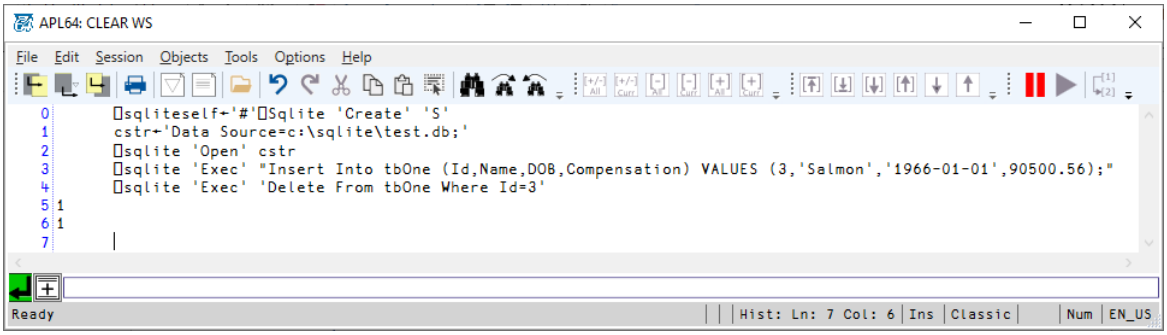
Syntax: [sqliteInstance] ⎕Sqlite 'Exec' cmd [subValue1] [subValue2] …

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

Example:

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'Exec' "Insert Into tbOne (Id,Name,DOB,Compensation) VALUES (3,'Salmon','1966-01-
01',90500.56);"
⎕sqlite 'Exec' 'Delete From tbOne Where Id=3'
```




## ExecDeleteQuery

Synonym: ExecDelete

Syntax: [sqliteInstance] ⎕Sqlite 'ExecDelete' cmd [subValue1] [subValue2] …

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

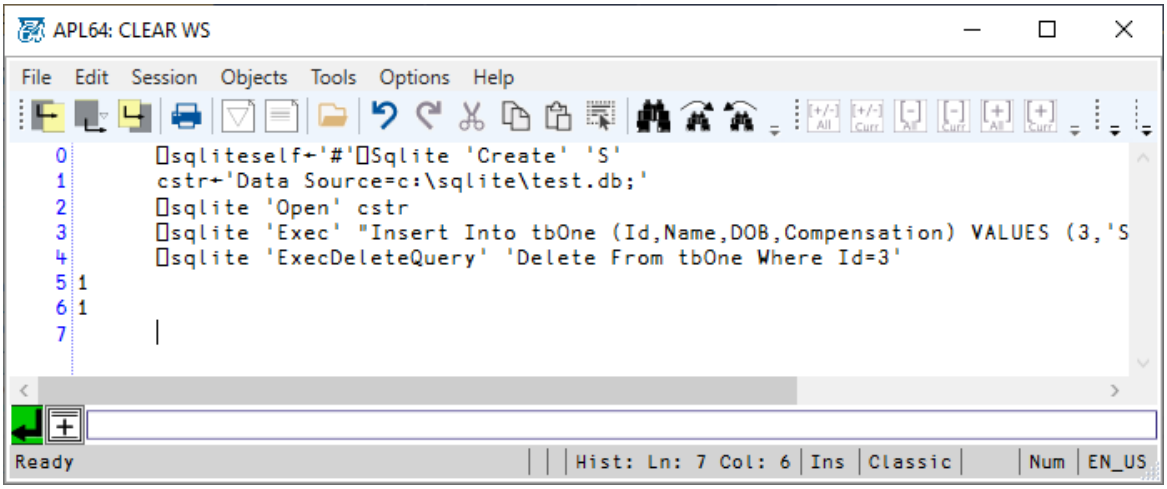
APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the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

Example:

```
⎕Sqlite 'ExecDelete' 'Delete From tbOne Where RcdId={0};' 3
```

Example:

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'Exec' "Insert Into tbOne (Id,Name,DOB,Compensation) VALUES (3,'Salmon','1966-01-01',90500.56);"
⎕sqlite 'ExecDeleteQuery' 'Delete From tbOne Where Id=3'
```




## ExecInsertQuery

Synonym: ExecInsert

Syntax: [sqliteInstance] ⎕sqlite 'ExecInsert'  tableName' colNames aplVals

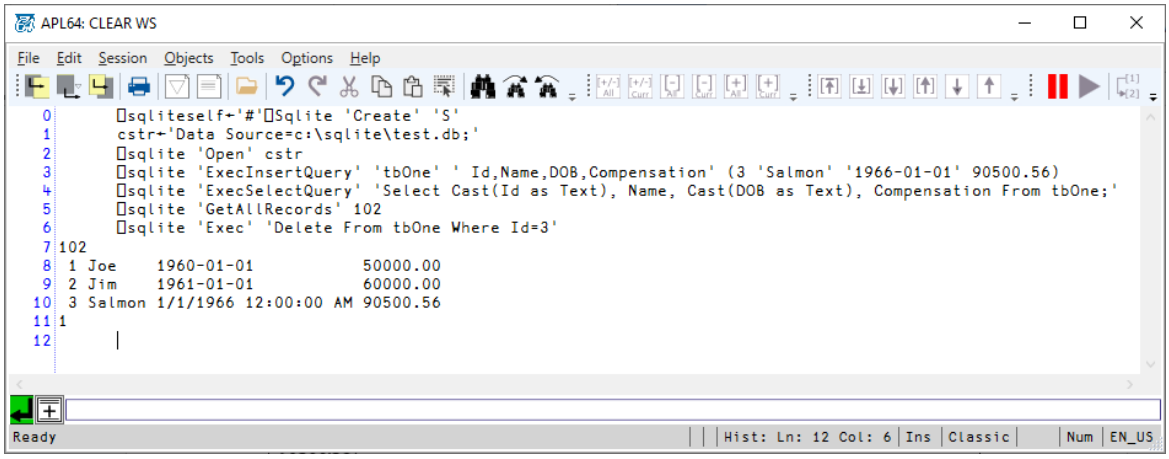
This action applies to an ⎕sqlite instance.
Right arguments:

1. Sqlite table name
2. Sqlite column names: Comma delimited text with selected Sqlite table column names, or ' ' or '*' to indicate all Sqlite 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 Sqlite column names and APL64 data matrix columns must be conformable.

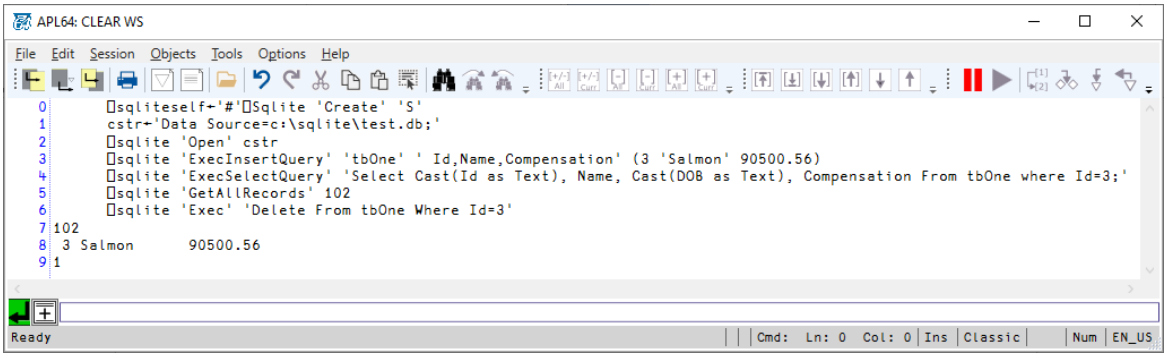
Specify all columns by name:

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecInsertQuery' 'tbOne' ' Id,Name,DOB,Compensation' (3 'Salmon' '1966-01-01' 90500.56)
⎕sqlite 'ExecSelectQuery' 'Select Cast(Id as Text), Name, Cast(DOB as Text), Compensation From tbOne;'
⎕sqlite 'GetAllRecords' 102
⎕sqlite 'Exec' 'Delete From tbOne Where Id=3'
```



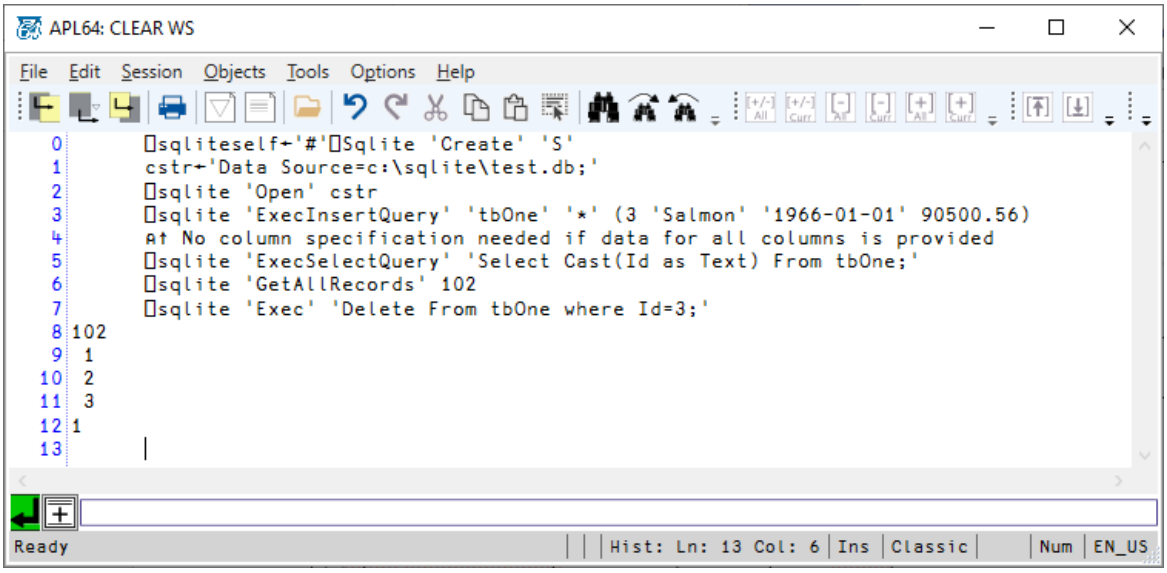

If only certain (nullable) column values are to be inserted, the required values for the non-nullable columns must also be inserted. Database null values are converted to APL64: 0 0⍴0:

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecInsertQuery' 'tbOne' ' Id,Name,Compensation' (3 'Salmon' 90500.56)
⎕sqlite 'ExecSelectQuery' 'Select Cast(Id as Text), Name, Cast(DOB as Text), Compensation From tbOne where Id=3;'
⎕sqlite 'GetAllRecords' 102
⎕sqlite 'Exec' 'Delete From tbOne Where Id=3'
```



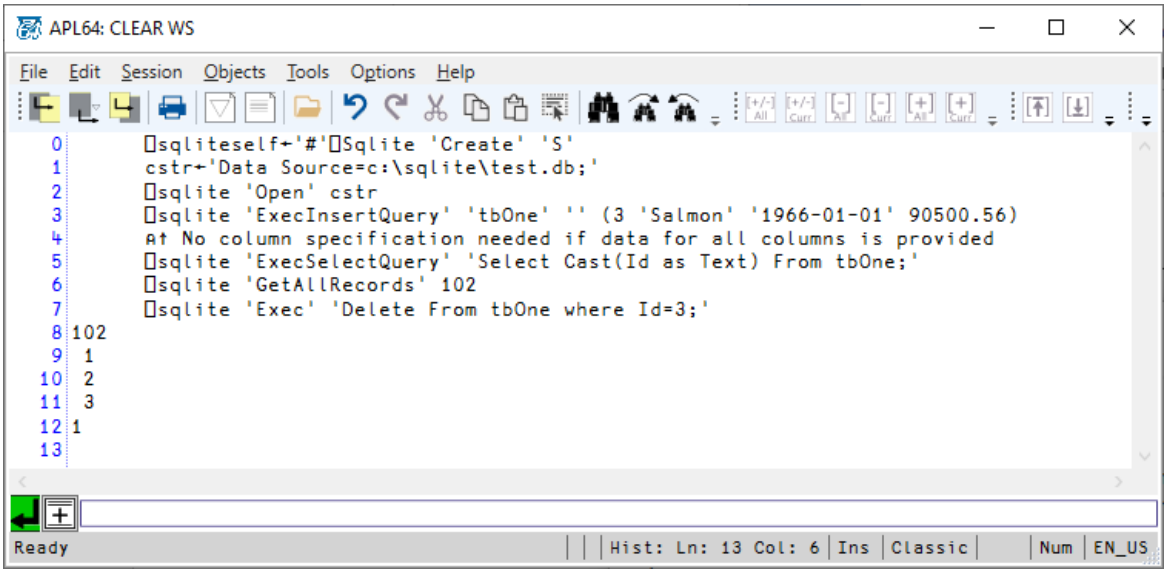

Specify all columns to be inserted with '*':

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecInsertQuery' 'tbOne' '*' (3 'Salmon' '1966-01-01' 90500.56)
⍝↑ No column specification needed if data for all columns is provided
⎕sqlite 'ExecSelectQuery' 'Select Cast(Id as Text) From tbOne;'
⎕sqlite 'GetAllRecords' 102
⎕sqlite 'Exec' 'Delete From tbOne where Id=3;'
```




No columns ('') need to be specified for the ExecInsertQuery, if the data to be inserted includes all columns:

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecInsertQuery' 'tbOne' '' (3 'Salmon' '1966-01-01' 90500.56)
⍝↑ No column specification needed if data for all columns is provided
⎕sqlite 'ExecSelectQuery' 'Select Cast(Id as Text) From tbOne;'
⎕sqlite 'GetAllRecords' 102
⎕sqlite 'Exec' 'Delete From tbOne where Id=3;'
```



## ExecSelectQuery

Synonym: ExecSelect

Syntax: [sqliteInstance] ⎕sqlite 'ExecSelect' cmd [subValue1] [subValue2] …

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

### Result is the Data Set Number

The ExecSelectQuery does not directly 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.

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

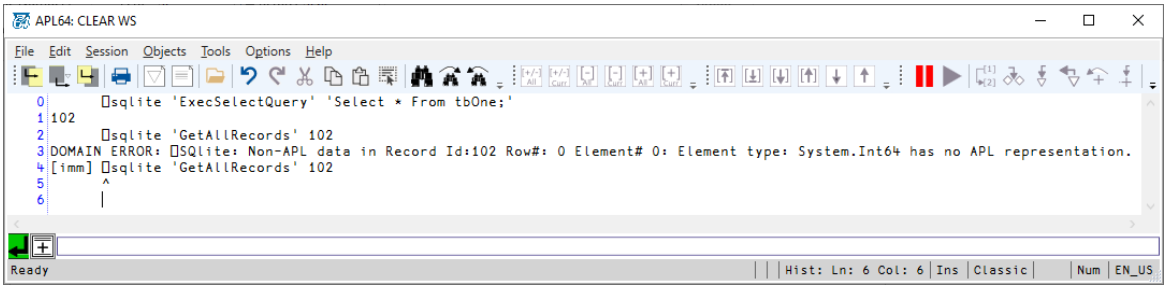
### One Record Set at a Time

The SQL-format query text argument to the ⎕SQLite 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.

```
⎕sqlite 'ExecSelectQuery' 'Select * From tbOne;'
⎕sqlite 'GetAllRecords' 102
```

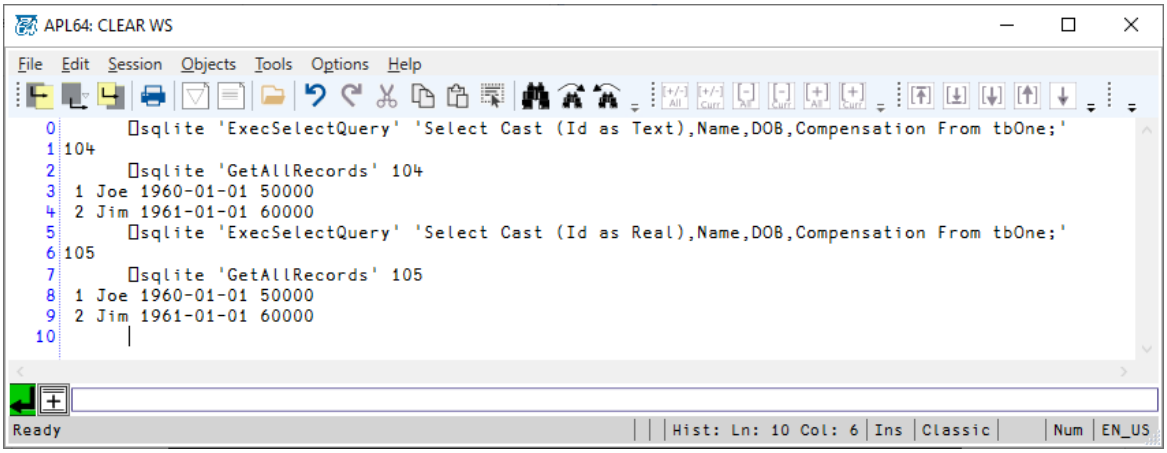
### Use Cast to handle non-APL Data Types

Executing the following APL statements will cause an exception because the Sqlite 'Integer' datatype is a 64-bit integer which has no representation in APL64:




To resolve this exception, the Cast function may be used:

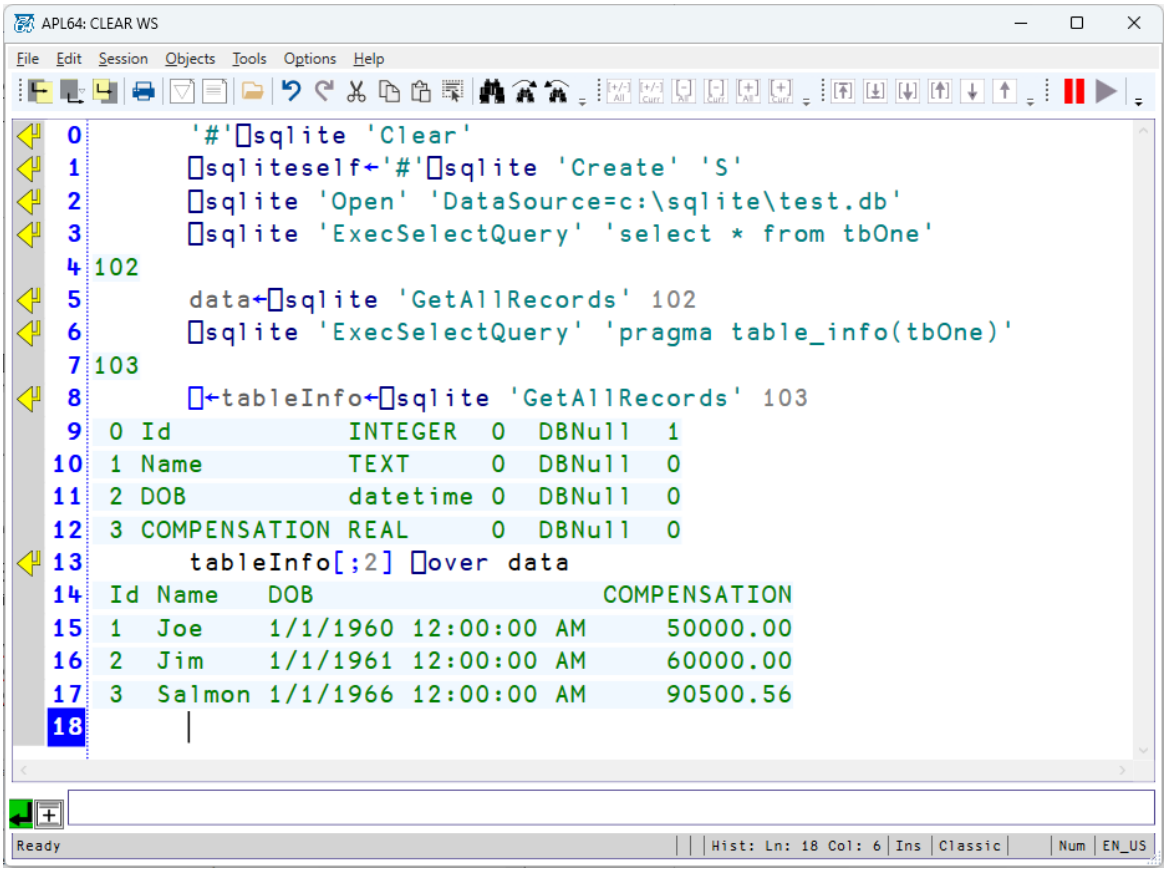
```
⎕sqlite 'ExecSelectQuery' 'Select Cast (Id as Text),Name,DOB,Compensation From tbOne;'
⎕sqlite 'GetAllRecords' 104
```



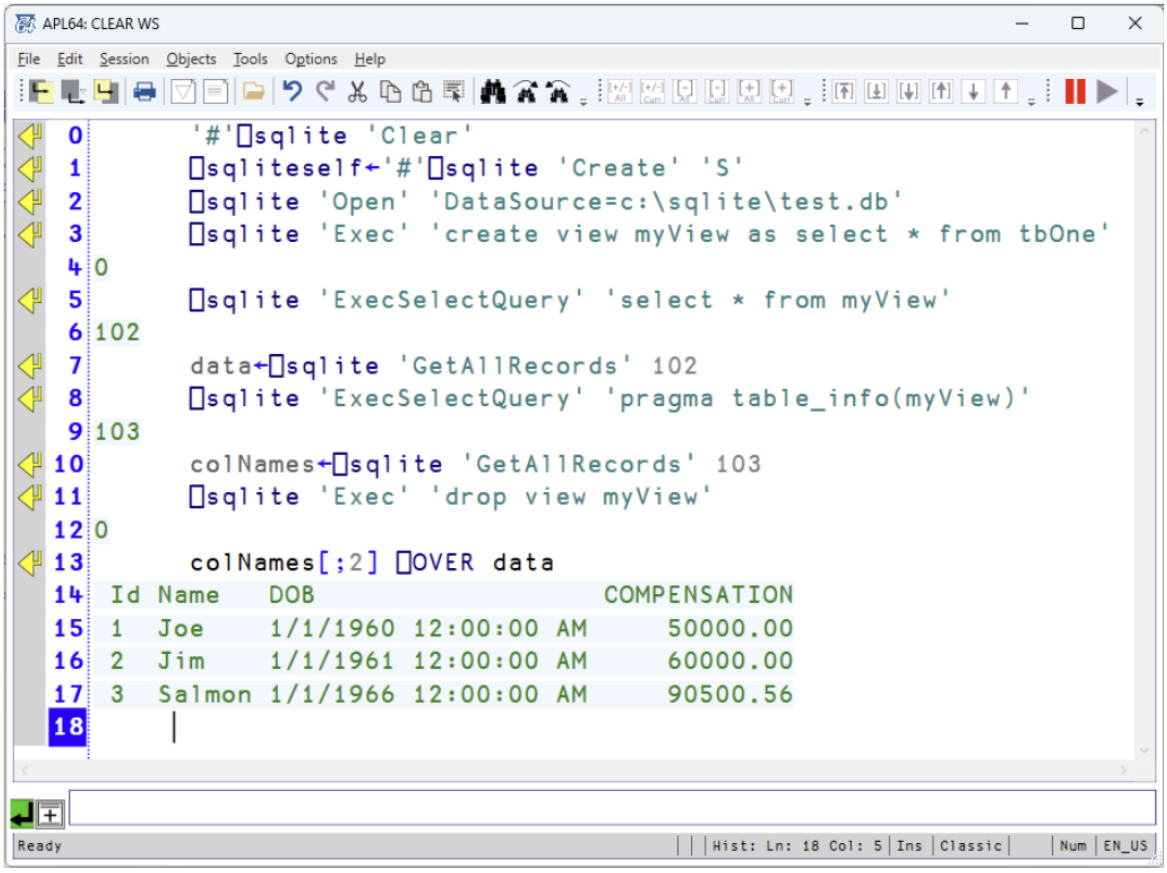

## Get Record Set Column Titles

Record set column titles are part of the table metadata. The Sqlite PRAGMA_TABLE_INFO() method can be used in a query to obtain this metadata.

Example of a record set which involves one table:




Example for a record set which may include multiple tables
When multiple tables are involved in a query, e.g. using join, the multiple-table metadata can be obtained by creating a view (virtual table) of the query. A view can also be used with a single table query.




## ExecStoredProc

Stored procedures are not supported by Sqlite.

## ExecStoredProcCmd

Stored procedures are not supported by Sqlite.

## GetAllRecords

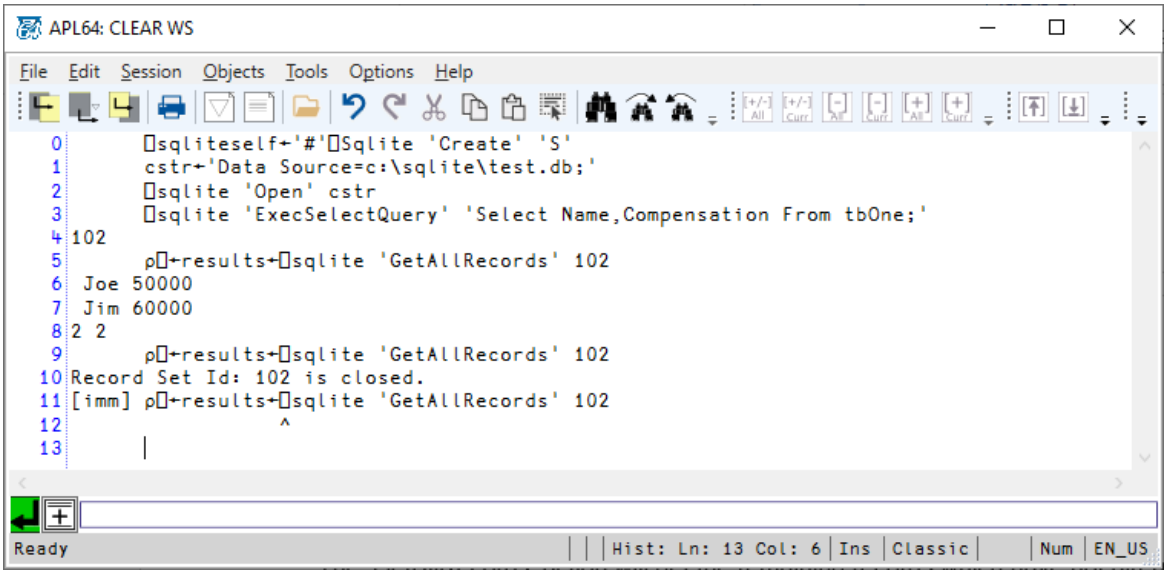
This action applies to an ⎕sqlite instance. The 'GetAllRecords' action will get the remaining records which have not already been read from the specified record set. This action closes the specified record set, because all records have been read.

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

An exception will 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.

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecSelectQuery' 'Select Name,Compensation From tbOne;'
```

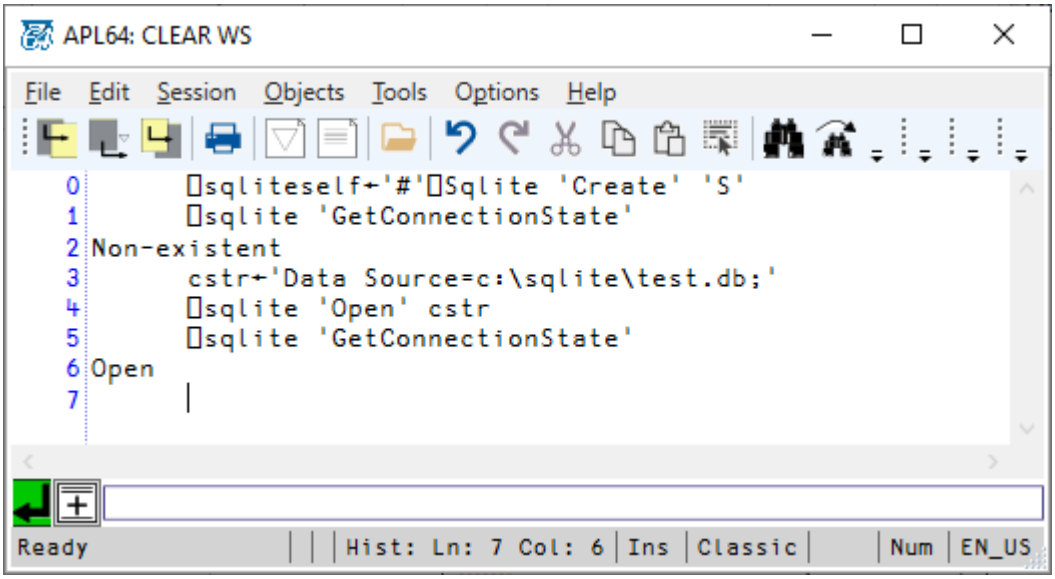
```
ρ⎕←results←⎕sqlite 'GetAllRecords' 102
ρ⎕←results←⎕sqlite 'GetAllRecords' 102
```




## GetConnectionState

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

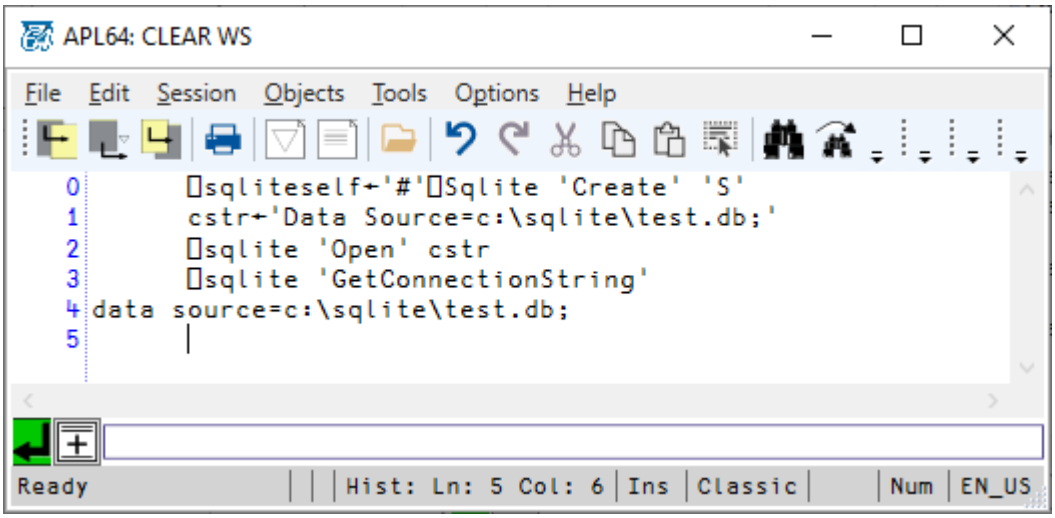
```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
⎕sqlite 'GetConnectionState'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'GetConnectionState'
```



## GetConnectionString

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

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
⎕sqlite 'GetConnectionString'
cstr←'Data                          Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'GetConnectionString'
```

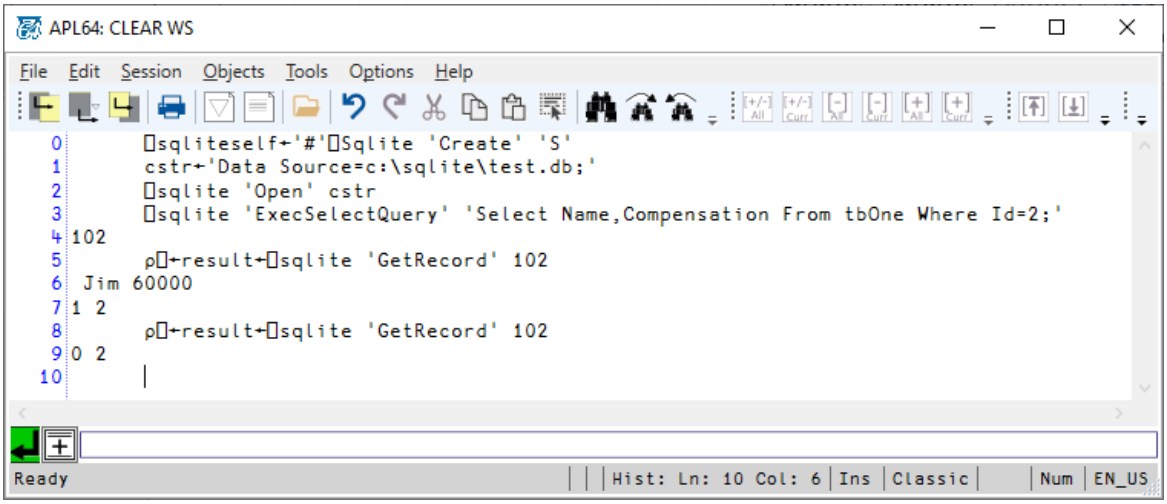



## GetRecord

This action applies to an ⎕sqlite instance. The right argument is the ⎕sqlite 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.

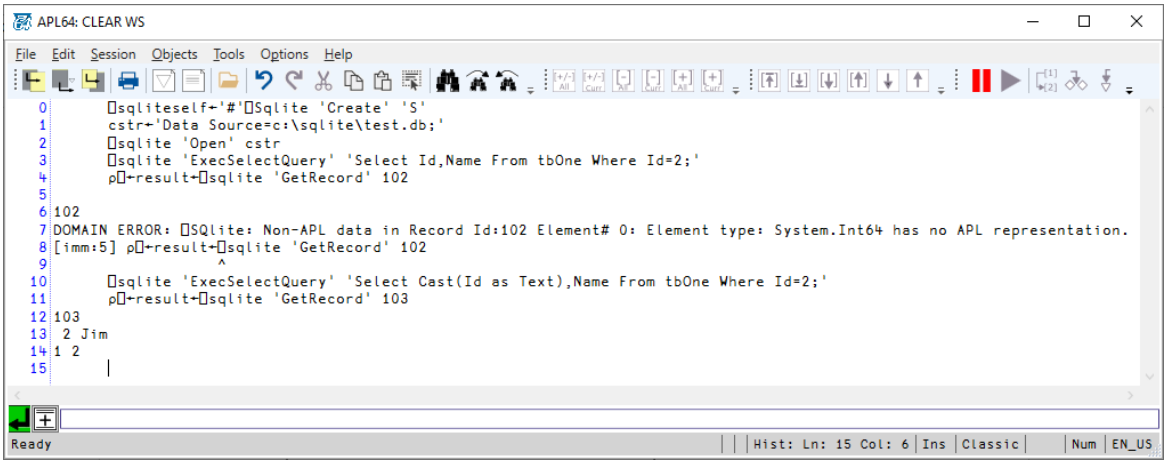
An exception will occur if the data in the record set has no representation in APL64. In this case the query which created the record set must be modified to make the returned data conformable with APL64.

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecSelectQuery' 'Select Name,Compensation From tbOne Where Id=2;'
ρ⎕←result←⎕sqlite 'GetRecord' 102
ρ⎕←result←⎕sqlite 'GetRecord' 102
```




An exception will 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. For example, in the sample database, the Id field is an Int64 integer:

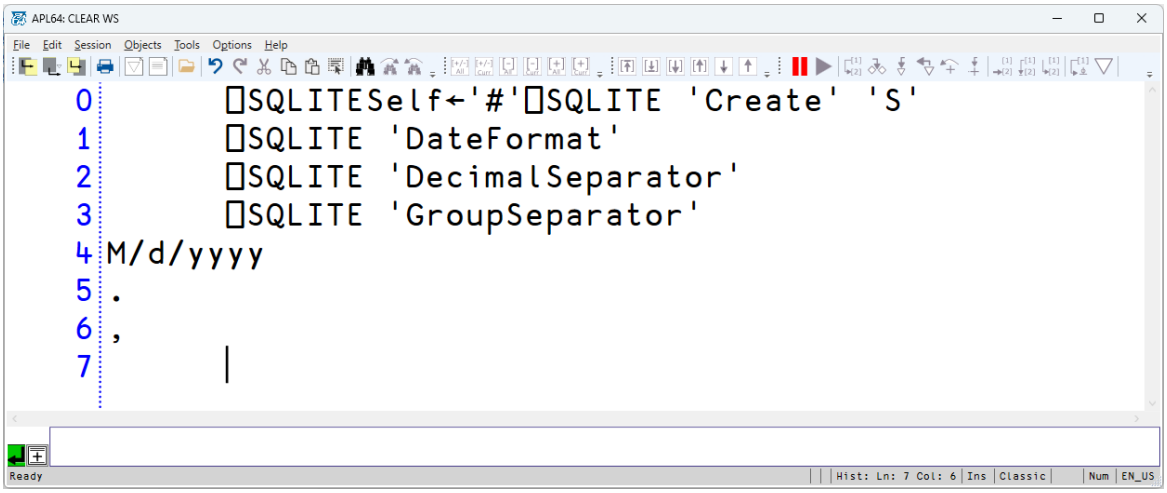
```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecSelectQuery' 'Select Id,Name From tbOne Where Id=2;'
⍴⎕←result←⎕sqlite 'GetRecord' 102
⎕sqlite 'ExecSelectQuery' 'Select Cast(Id as Text),Name From tbOne Where Id=2;'
⍴⎕←result←⎕sqlite 'GetRecord' 103
```




## GroupSeparator

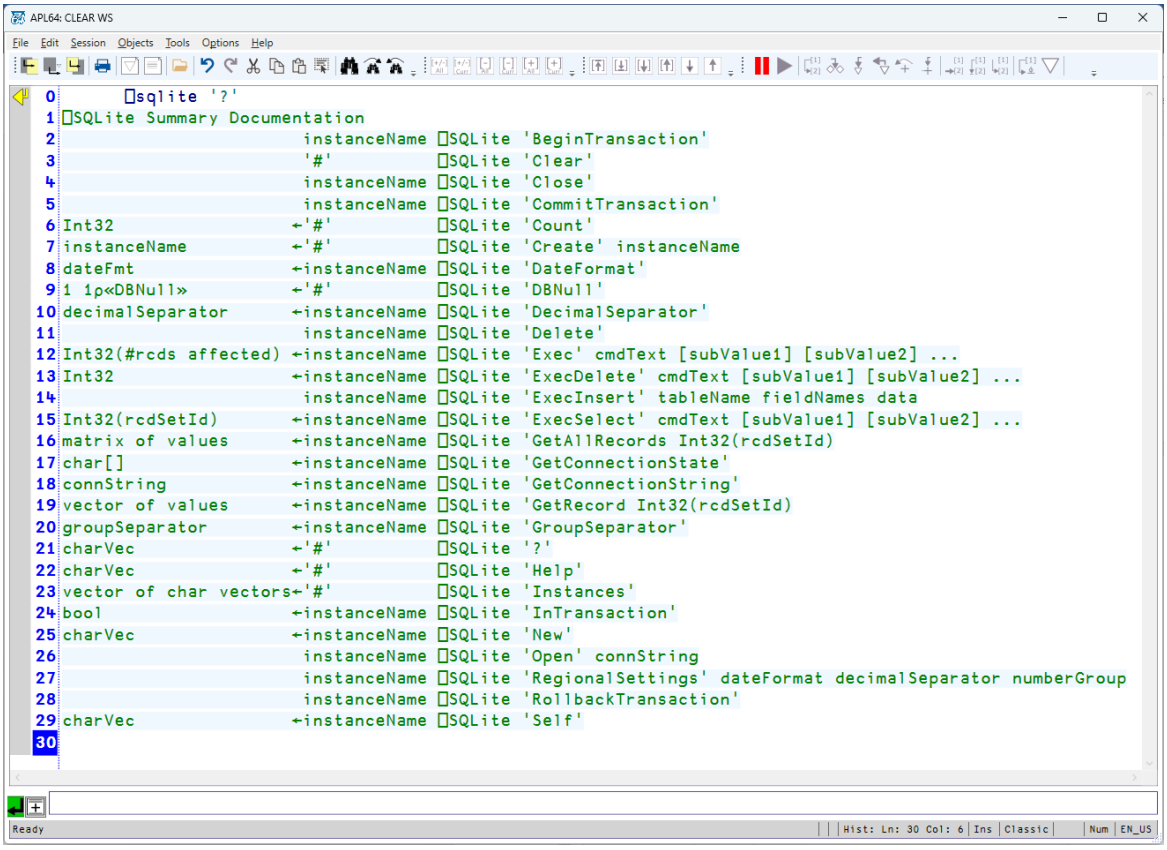
This action returns the .Net Number Group Separator text applicable to the specified ⎕SQLMY instance.

```
⎕SQLMYSelf←'#'⎕SQLMY 'Create' 'S'
⎕SQLMY 'DateFormat'
⎕SQLMY 'DecimalSeparator'
⎕SQLMY 'GroupSeparator'
```




## ? or Help

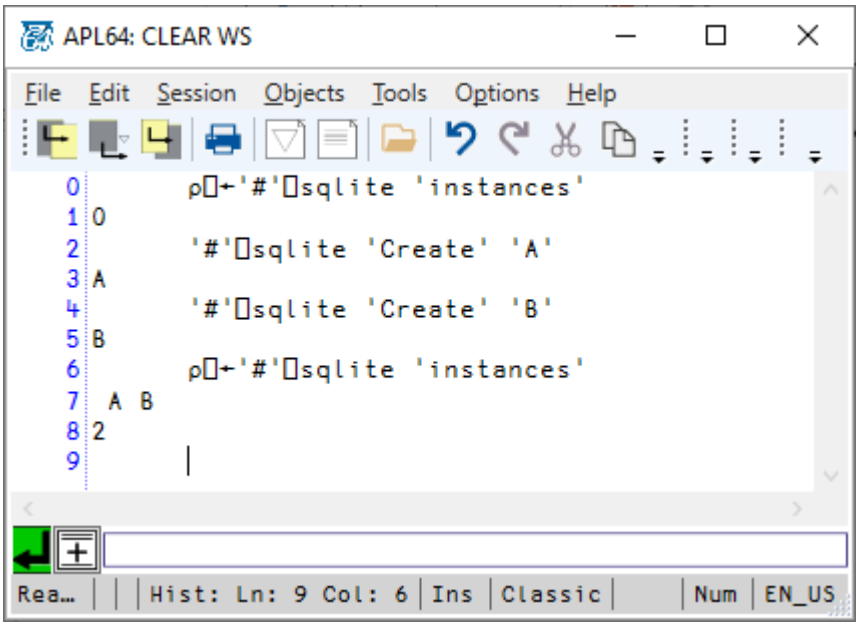
This action is performed on the ⎕sqlite object. The ? and Help actions will return a text array containing a summary of the ⎕sqlite syntax.



## Instances

This action is performed on the ⎕sqlite object. The result of the Instances action is the number of existing ⎕sqlite instances in the APL64 instance.

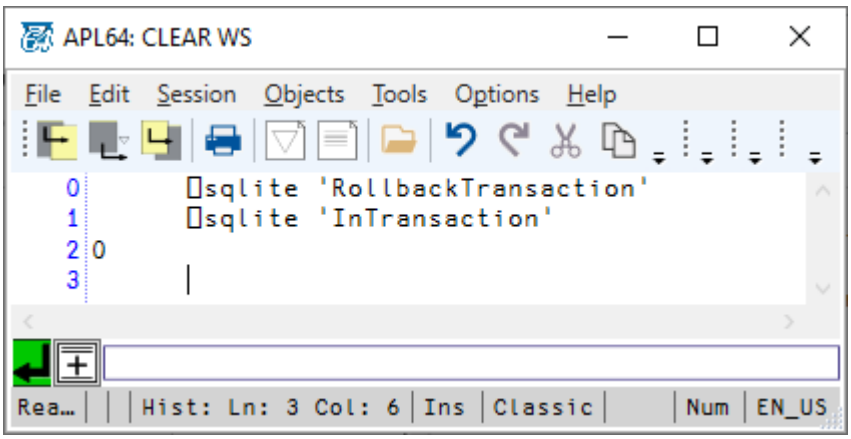
```
'#'⎕sqlite 'Clear'
ρ⎕←'#'⎕sqlite 'instances'
'#'⎕sqlite 'Create' 'A'
'#'⎕sqlite 'Create' 'B'
ρ⎕←'#'⎕sqlite 'instances'
```




## InTransaction

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

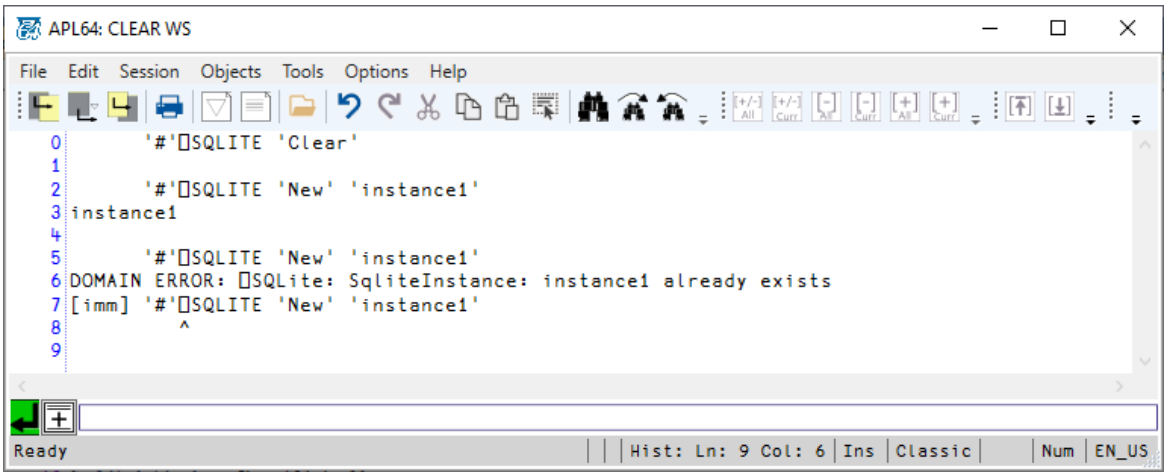
```
⎕sqlite 'RollbackTransaction'
⎕sqlite 'InTransaction'
```



## New

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

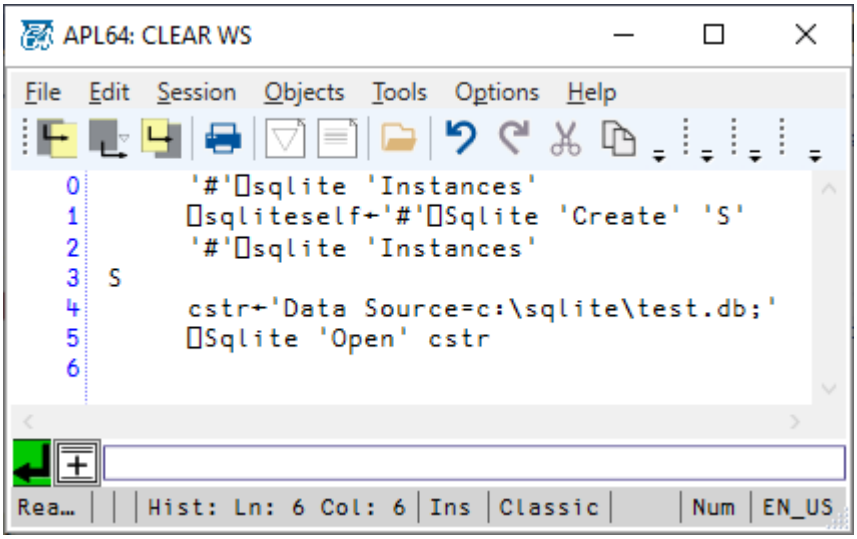
```
'#'⎕SQLITE 'Clear'
'#'⎕SQLITE 'New' 'instance1'
'#'⎕SQLITE 'New' 'instance1'
```




## Open

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

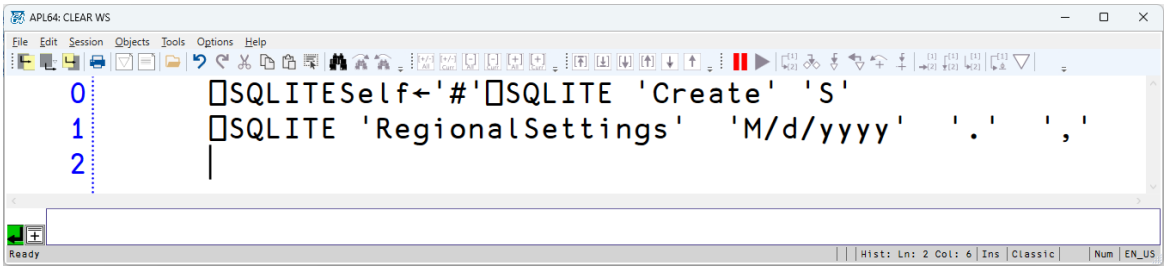
```
'#'⎕sqlite 'Instances'
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
'#'⎕sqlite 'Instances'
cstr←'Data Source=c:\sqlite\test.db;'
⎕Sqlite 'Open' cstr
```




## RegionalSettings

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

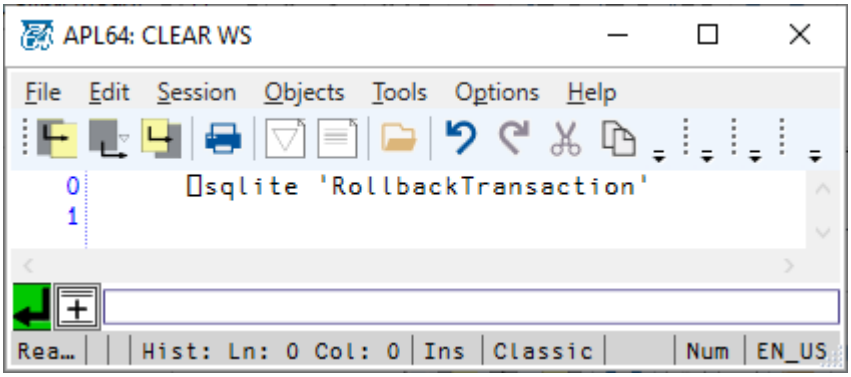
```
⎕SQLITESelf←'#'⎕SQLITE 'Create' 'S'
⎕SQLITE 'RegionalSettings' 'M/d/yyyy' '.' ','
```



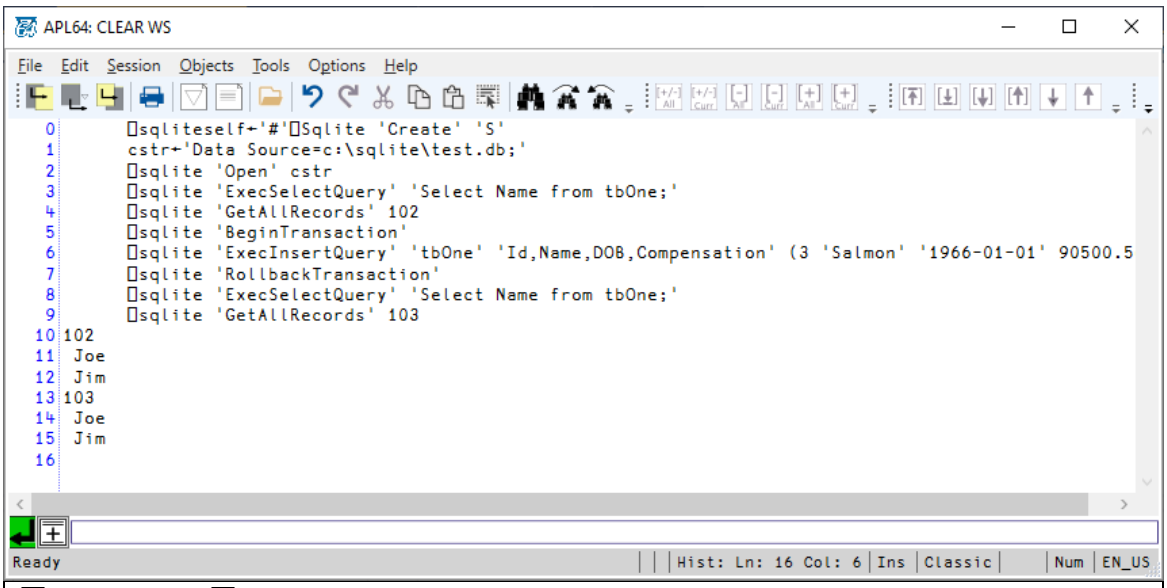

## RollbackTransaction

This action applies to an ⎕sqlite 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.

```
⎕sqlite 'RollbackTransaction
```



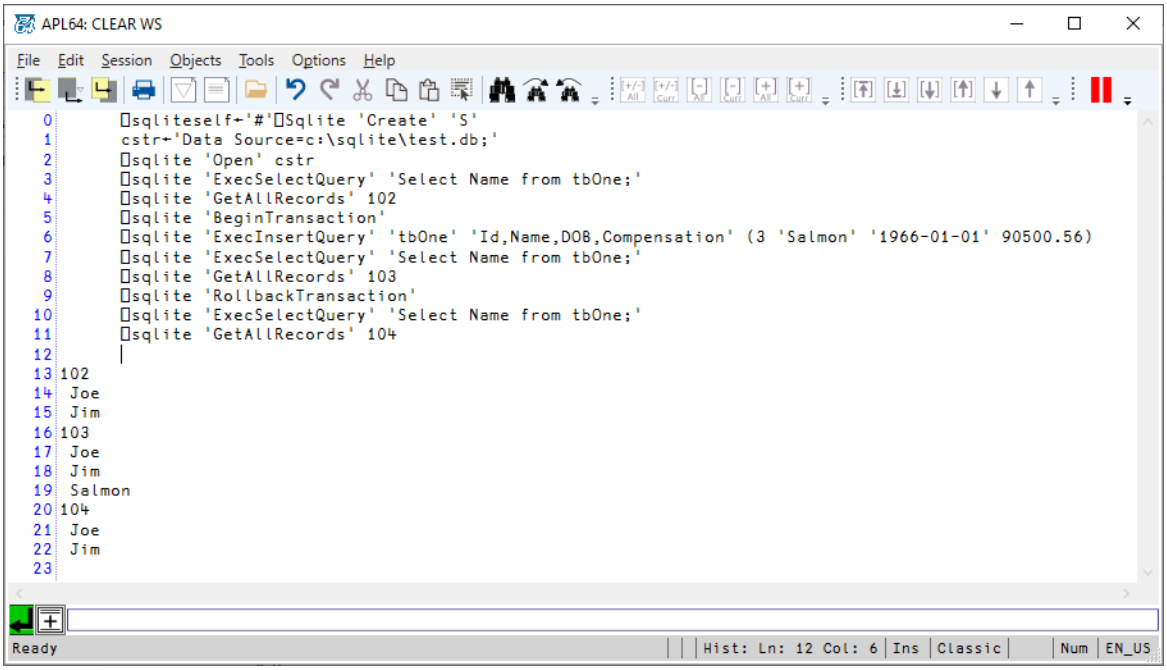

```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
⎕sqlite 'GetAllRecords' 102
⎕sqlite 'BeginTransaction'
⎕sqlite 'ExecInsertQuery' 'tbOne' 'Id,Name,DOB,Compensation' (3 'Salmon' '1966-01-01' 90500.56)
⎕sqlite 'RollbackTransaction'
⎕sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
⎕sqlite 'GetAllRecords' 103
```




```
⎕sqliteself←'#'⎕Sqlite 'Create' 'S'
cstr←'Data Source=c:\sqlite\test.db;'
⎕sqlite 'Open' cstr
⎕sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
⎕sqlite 'GetAllRecords' 102
⎕sqlite 'BeginTransaction'
⎕sqlite 'ExecInsertQuery' 'tbOne' 'Id,Name,DOB,Compensation' (3 'Salmon' '1966-01-01' 90500.56)
```

```
□sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
□sqlite 'GetAllRecords' 103
□sqlite 'RollbackTransaction'
□sqlite 'ExecSelectQuery' 'Select Name from tbOne;'
□sqlite 'GetAllRecords' 104
```

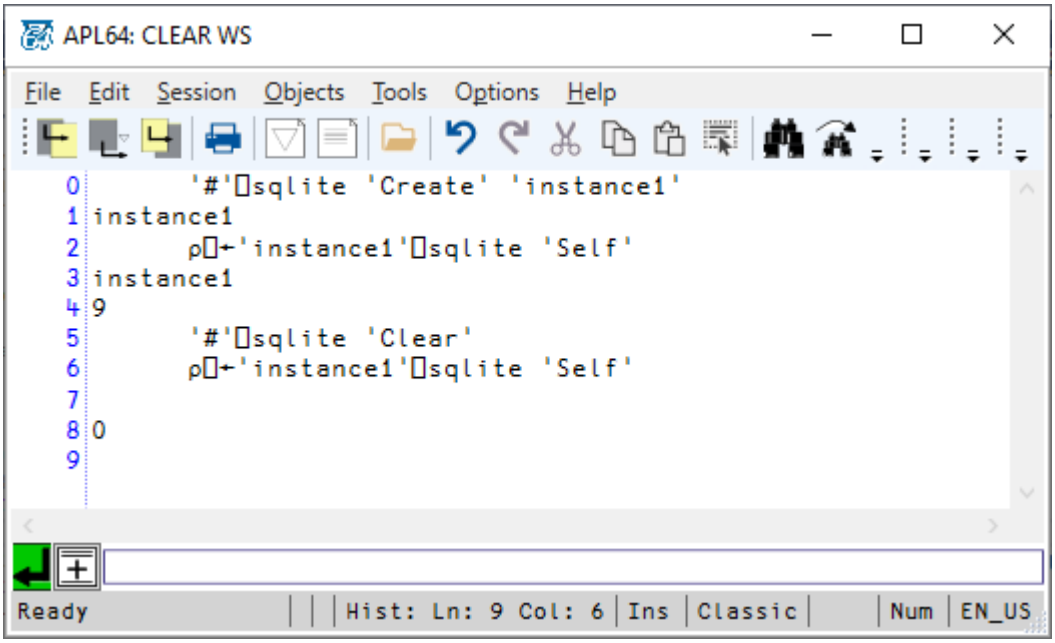
Note that the records in the #103 data set include the pending effects of the ExecInsertQuery action, but the records in the #104 data set do not include the effects of the rollbacked ExecInsertQuery action:




## Self

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

```
'#'□sqlite 'Create' 'instance1'
ρ□←'instance1'□sqlite 'Self'
'#'□sqlite 'Clear'
ρ□←'instance1'□sqlite 'Self'
```




## Learn Structured Query Language

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