Description

`odbc` allows you to load, write, and view data from Open DataBase Connectivity (ODBC) sources into Stata. ODBC is a standardized set of function calls for accessing data stored in both relational and nonrelational database-management systems. By default on Unix platforms, `iODBC` is the ODBC driver manager Stata uses, but you can use `unixODBC` by using the command `set odbcmgr unixodbc`.

ODBC’s architecture consists of four major components (or layers): the client interface, the ODBC driver manager, the ODBC drivers, and the data sources. Stata provides `odbc` as the client interface. The system is illustrated as follows:

- `odbc list` produces a list of ODBC data source names to which Stata can connect.
- `odbc query` retrieves a list of table names available from a specified data source’s system catalog.
- `odbc describe` lists column names and types associated with a specified table.
- `odbc load` reads an ODBC table into memory. You can load an ODBC table specified in the `table()` option or load an ODBC table generated by an SQL SELECT statement specified in the `exec()` option. In both cases, you can choose which columns and rows of the ODBC table to read by specifying `extvarlist` and `if` and `in` conditions. `extvarlist` specifies the columns to be read and allows you to rename variables. For example,

  ```
  . odbc load id=ID name="Last Name", table(Employees) dsn(Northwind)
  ```

  reads two columns, `ID` and `Last Name`, from the `Employees` table of the `Northwind` data source. It will also rename variable `ID` to `id` and variable `Last Name` to `name`.

- `odbc insert` writes data from memory to an ODBC table. The data can be appended to an existing table or replace an existing table.

- `odbc exec` allows for most SQL statements to be issued directly to any ODBC data source. Statements that produce output, such as `SELECT`, have their output neatly displayed. By using Stata’s ado language, you can also generate SQL commands on the fly to do positional updates or whatever the situation requires.

- `odbc sqlfile` provides a “batch job” alternative to the `odbc exec` command. A file is specified that contains any number of any length SQL commands. Every SQL command in this file should be delimited by a semicolon and must be constructed as pure SQL. Stata macros and ado-language syntax
are not permitted. The advantage in using this command, as opposed to `odbc exec`, is that only one connection is established for multiple SQL statements. A similar sequence of SQL commands used via `odbc exec` would require constructing an ado-file that issued a command and, thus, a connection for every SQL command. Another slight difference is that any output that might be generated from an SQL command is suppressed by default. A `loud` option is provided to toggle output back on.

`set odbcdriver unicode` specifies that the ODBC driver is a Unicode driver (the default). `set odbcdriver ansi` specifies that the ODBC driver is an ANSI driver. You must restart Stata for the setting to take effect.

`set odbcmgr iodbc` specifies that the ODBC driver manager is iODBC (the default). `set odbcmgr unixodbc` specifies that the ODBC driver manager is unixODBC.

**Quick start**

List all defined data source names (DSNs) to which Stata can connect

```
    odbc list
```

List available table names in `MyDSN`

```
    odbc query "MyDSN"
```

Describe the column names and data types in table `MyTable` from `MyDSN`

```
    odbc describe "MyTable", dsn("MyDSN")
```

Load `MyTable` into memory from `MyDSN`

```
    odbc load, table("MyTable") dsn("MyDSN")
```

**Menu**

```
odbc load
File > Import > ODBC data source
```

```
odbc insert
File > Export > ODBC data source
```
Syntax

List ODBC sources to which Stata can connect

```
odbc list
```

Retrieve available names from specified data source

```
odbc query ["DataSourceName", verbose schema connect_options]
```

List column names and types associated with specified table

```
odbc describe ["TableName", connect_options]
```

Import data from an ODBC data source

```
odbc load [extvarlist] [if] [in], \{table("TableName")|exec("SqlStmt")\}
[load_options connect_options]
```

Export data to an ODBC data source

```
odbc insert [varlist] [if] [in], table("TableName")
{dsn("DataSourceName")|connectionstring("ConnectStr")}
[insert_options connect_options]
```

Allow SQL statements to be issued directly to ODBC data source

```
odbc exec("SqlStmt") ,
{dsn("DataSourceName")|connectionstring("ConnectStr")}
[connect_options]
```

Batch job alternative to odbc exec

```
odbc sqlfile("filename") ,
{dsn("DataSourceName")|connectionstring("ConnectStr")}
[loud connect_options]
```

Specify ODBC driver type

```
set odbcdriver {unicode|ansi} [, permanently]
```

Specify ODBC driver manager (Mac and Unix only)

```
set odbcmgr {iodbc|unixodbc} [, permanently]
```
**DataSourceName** is the name of the ODBC source (database, spreadsheet, etc.)

**ConnectStr** is a valid ODBC connection string

**TableName** is the name of a table within the ODBC data source

**SqlStmt** is an SQL SELECT statement

**filename** is pure SQL commands separated by semicolons

**extvarlist** contains

- `sqlvarname`  
  - `varname = sqlvarname`

### connect_options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>user(UserID)</code></td>
<td>user ID of user establishing connection</td>
</tr>
<tr>
<td><code>password(Password)</code></td>
<td>password of user establishing connection</td>
</tr>
<tr>
<td><code>dialog(noprompt)</code></td>
<td>do not display ODBC connection-information dialog, and</td>
</tr>
<tr>
<td></td>
<td>do not prompt user for connection information</td>
</tr>
<tr>
<td><code>dialog(prompt)</code></td>
<td>display ODBC connection-information dialog</td>
</tr>
<tr>
<td><code>dialog(complete)</code></td>
<td>display ODBC connection-information dialog only if there is not enough</td>
</tr>
<tr>
<td></td>
<td>information held</td>
</tr>
<tr>
<td><code>dialog(required)</code></td>
<td>display ODBC connection-information dialog only if there is not enough</td>
</tr>
<tr>
<td></td>
<td>mandatory information provided</td>
</tr>
<tr>
<td><code>dsn(&quot;DataSourceName&quot;)</code></td>
<td>name of data source</td>
</tr>
<tr>
<td><code>connectionstring(&quot;ConnectStr&quot;)</code></td>
<td>ODBC connection string</td>
</tr>
</tbody>
</table>

* `dsn("DataSourceName")` is not allowed with odbc query. You may not specify both **DataSourceName** and **connectionstring()** with odbc query. Either `dsn()` or `connectionstring()` is required with odbc insert, odbc exec, and odbc sqlfile.

### load_options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>table(&quot;TableName&quot;)</code></td>
<td>name of table stored in data source</td>
</tr>
<tr>
<td><code>exec(&quot;SqlStmt&quot;)</code></td>
<td>SQL SELECT statement to generate a table to be read into Stata</td>
</tr>
<tr>
<td><code>clear</code></td>
<td>load dataset even if there is one in memory</td>
</tr>
<tr>
<td><code>noquote</code></td>
<td>alter Stata’s internal use of SQL commands; seldom used</td>
</tr>
<tr>
<td><code>lowercase</code></td>
<td>read variable names as lowercase</td>
</tr>
<tr>
<td><code>sqlshow</code></td>
<td>show all SQL commands issued</td>
</tr>
<tr>
<td><code>allstring</code></td>
<td>read all variables as strings</td>
</tr>
<tr>
<td><code>datestring</code></td>
<td>read date-formatted variables as strings</td>
</tr>
<tr>
<td><code>multistatement</code></td>
<td>allow multiple SQL statements delimited by ; when using <code>exec()</code></td>
</tr>
<tr>
<td><code>bigintasdouble</code></td>
<td>store BIGINT columns as Stata doubles on 64-bit operating systems</td>
</tr>
</tbody>
</table>

* Either `table("TableName")` or `exec("SqlStmt")` must be specified with odbc load.
**insert_options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>table(&quot;TableName&quot;)</em></td>
<td>name of table stored in data source</td>
</tr>
<tr>
<td>overwrite</td>
<td>clear data in ODBC table before data in memory is written to the table</td>
</tr>
<tr>
<td>insert</td>
<td>default mode of operation for the <code>odbc insert</code> command</td>
</tr>
<tr>
<td>quoted</td>
<td>quote all values with single quotes as they are inserted in ODBC table</td>
</tr>
<tr>
<td>sqlshow</td>
<td>show all SQL commands issued</td>
</tr>
<tr>
<td>as(&quot;varlist&quot;)</td>
<td>ODBC variables on the data source that correspond to the variables in Stata’s memory</td>
</tr>
<tr>
<td>block</td>
<td>use block inserts</td>
</tr>
</tbody>
</table>

*`table("TableName")`* is required for `odbc insert`.

### Options

`user(UserID)` specifies the user ID of the user attempting to establish the connection to the data source. By default, Stata assumes that the user ID is the same as the one specified in the previous `odbc` command or is empty if `user()` has never been specified in the current session of Stata.

`password(Password)` specifies the password of the user attempting to establish the connection to the data source. By default, Stata assumes that the password is the same as the one previously specified or is empty if the password has not been used during the current session of Stata. Typically, the `password()` option will not be specified apart from the `user()` option.

`dialog(noprompt | prompt | complete | required)` specifies the mode the ODBC Driver Manager uses to display the ODBC connection-information dialog to prompt for more connection information.

`noprompt` is the default value. The ODBC connection-information dialog is not displayed, and you are not prompted for connection information. If there is not enough information to establish a connection to the specified data source, an error is returned.

`prompt` causes the ODBC connection-information dialog to be displayed.

`complete` causes the ODBC connection-information dialog to be displayed only if there is not enough information, even if the information is not mandatory.

`required` causes the ODBC connection-information dialog to be displayed only if there is not enough mandatory information provided to establish a connection to the specified data source. You are prompted only for mandatory information; controls for information that is not required to connect to the specified data source are disabled.

`dsn("DataSourceName")` specifies the name of a data source, as listed by the `odbc list` command. If a name contains spaces, it must be enclosed in double quotes. By default, Stata assumes that the data source name is the same as the one specified in the previous `odbc` command. This option is not allowed with `odbc query`. Either the `dsn()` option or the `connectionstring()` option may be specified with `odbc describe` and `odbc load`, and one of these options must be specified with `odbc insert`, `odbc exec`, and `odbc sqlfile`.

`connectionstring("ConnectStr")` specifies a connection string rather than the name of a data source. Stata does not assume that the connection string is the same as the one specified in the previous `odbc` command. Either `DataSourceName` or the `connectionstring()` option may be specified with `odbc query`; either the `dsn()` option or the `connectionstring()` option can be specified with `odbc describe` and `odbc load`, and one of these options must be specified with `odbc insert`, `odbc exec`, and `odbc sqlfile`. 
table("TableName") specifies the name of an ODBC table stored in a specified data source’s system catalog, as listed by the `odbc query` command. If a table name contains spaces, it must be enclosed in double quotes. Either the `table()` option or the `exec()` option—but not both—is required with the `odbc load` command.

`exec("SqlStmt")` allows you to issue an SQL SELECT statement to generate a table to be read into Stata. An error message is returned if the SELECT statement is an invalid SQL statement. The statement must be enclosed in double quotes. Either the `table()` option or the `exec()` option—but not both—is required with the `odbc load` command.

clear permits the data to be loaded, even if there is a dataset already in memory, and even if that dataset has changed since the data were last saved.

`noquote` alters Stata’s internal use of SQL commands, specifically those relating to quoted table names, to better accommodate various drivers. This option has been particularly helpful for DB2 drivers.

`lowercase` causes all the variable names to be read as lowercase.

`sqlshow` is a useful option for showing all SQL commands issued to the ODBC data source from the `odbc insert` or `odbc load` command. This can help you debug any issues related to inserting or loading.

`allstring` causes all variables to be read as string data types.

`datestring` causes all date- and time-formatted variables to be read as string data types.

`multistatement` specifies that multiple SQL statements delimited by `;` be allowed when using the `exec()` option. Some drivers do not support multiple SQL statements.

`bigintastdouble` specifies that data stored in 64-bit integer (BIGINT) database columns be converted to Stata doubles. If any integer value is larger than 9,007,199,254,740,965 or less than −9,007,199,254,740,992, this conversion is not possible, and `odbc load` will issue an error message.

`overwrite` allows data to be cleared from an ODBC table before the data in memory are written to the table. All data from the ODBC table are erased, not just the data from the variable columns that will be replaced.

`insert` appends data to an existing ODBC table and is the default mode of operation for the `odbc insert` command.

`quoted` is useful for ODBC data sources that require all inserted values to be quoted. This option specifies that all values be quoted with single quotes as they are inserted into an ODBC table.

`as("varlist")` allows you to specify the ODBC variables on the data source that correspond to the variables in Stata’s memory. If this option is specified, the number of variables must equal the number of variables being inserted, even if some names are identical.

`loud` specifies that output be displayed for SQL commands.

`verbose` specifies that `odbc query list` any data source alias, nickname, typed table, typed view, and view along with tables so that you can load data from these table types.

`schema` specifies that `odbc query return` schema names with the table names from a data source. Note: The schema names returned from `odbc query` will also be used with the `odbc describe` and `odbc load` commands. When using `odbc load` with a schema name, you might also need to specify the `noquote` option because some drivers do not accept quotes around table or schema names.
block specifies that odbc insert use block inserts to speed up data-writing performance. Some drivers do not support block inserts.

permanently (set odbcdriver and set odbcmgr only) specifies that, in addition to making the change right now, the setting be remembered and become the default setting when you invoke Stata.

Remarks and examples

When possible, the examples in this manual entry are developed using the Northwind sample database that is automatically installed with Microsoft Access. If you do not have Access, you can still use odbc, but you will need to consult the documentation for your other ODBC sources to determine how to set them up.

Remarks are presented under the following headings:

- Unicode and ODBC
- Setting up the data sources
- Listing ODBC data source names
- Listing available table names from a specified data source’s system catalog
- Describing a specified table
- Loading data from ODBC sources

Unicode and ODBC

Stata supports accessing databases with Unicode data through Unicode ODBC drivers on the following platforms:

- Microsoft Windows through ODBC driver manager (version 3.5 or higher).
- Unix through unixODBC driver manager with ODBC drivers compiled for unixODBC. Stata does not support Unicode drivers when using iodbc as your driver manager. Stata requires that the driver support UTF-8.
- macOS through unixODBC driver manager with ODBC drivers compiled for unixODBC. Stata does not support Unicode drivers when using iodbc as your driver manager. Stata requires that the driver support UTF-8.

Stata supports non-Unicode databases through ASCII drivers with all driver managers.

Setting up the data sources

Before using Stata’s ODBC commands, you must register your ODBC database with the ODBC Data Source Administrator. This process varies depending on platform, but the following example shows the steps necessary for Windows.

Using Windows 10, follow these steps to create an ODBC User Data Source for the Northwind sample database:

1. On the Start page, type ODBC Data Sources. From the list that appears, select the ODBC Data Sources Desktop App.
2. In the Data Sources (ODBC) dialog box,
   a. click on the User DSN tab;
   b. click on Add...;
c. choose Microsoft Access Driver (*.mdb, *.accdb) on the Create New Data Source dialog box; and

d. click on Finish.

3. In the ODBC Microsoft Access Setup dialog box, type Northwind in the Data Source Name field and click on Select. Locate the Northwind.mdb database and click on OK to finish creating the data source.

Using Windows 7, follow these steps to create an ODBC User Data Source for the Northwind sample database:

1. From the Start Menu, select the Control Panel.
2. In the Control Panel window, click on System and Security > Administrative Tools.
3. In the Data Sources (ODBC) dialog box,
   a. click on the User DSN tab;
   b. click on Add...;
   c. choose Microsoft Access Driver (*.mdb, *.accdb) on the Create New Data Source dialog box; and
   d. click on Finish.
4. In the ODBC Microsoft Access Setup dialog box, type Northwind in the Data Source Name field and click on Select. Locate the Northwind.mdb database and click on OK to finish creating the data source.

Technical note

In earlier versions of Windows, the exact location of the Data Source (ODBC) dialog varies, but it is always somewhere within the Control Panel.

Listing ODBC data source names

odbc list is used to produce a list of data source names to which Stata can connect. For a specific data source name to be shown in the list, the data source has to be registered with the ODBC Data Source Administrator. See Setting up the data sources for information on how to do this.

Example 1

```
. odbc list
Data Source Name       Driver
     dBASE Files          Microsoft Access dBASE Driver (*.dbf, *.ndx
     Excel Files          Microsoft Excel Driver (*.xls, *.xlsx, *.xl
     MS Access Database   Microsoft Access Driver (*.mdb, *.accdb)
    Northwind            Microsoft Access Driver (*.mdb, *.accdb)
```

In the above list, Northwind is one of the sample Microsoft Access databases that Access installs by default.
Listing available table names from a specified data source’s system catalog

odbc query is used to list table names available from a specified data source.

Example 2

    . odbc query "Northwind"
    DataSource: Northwind
    Path : C:\Program Files\Microsoft Office\Office\Samples\Northwind.accdb

    Customers
    Employee Privileges
    Employees
    Inventory Transaction Types
    Inventory Transactions
    Invoices
    Order Details
    Order Details Status
    Orders
    Orders Status
    Orders Tax Status
    Privileges
    Products
    Purchase Order Details
    Purchase Order Status
    Purchase Orders
    Sales Reports
    Shippers
    Strings
    Suppliers
Describing a specified table

`odbc describe` is used to list column (variable) names and their SQL data types that are associated with a specified table.

➢ Example 3

Here we specify that we want to list all variables in the `Employees` table of the `Northwind` data source.

```
. odbc describe "Employees", dsn("Northwind")
```

Table: Employees (load)

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>COUNTER</td>
</tr>
<tr>
<td>Company</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Last Name</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>First Name</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>E-mail Address</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Job Title</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Business Phone</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Home Phone</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Mobile Phone</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Fax Number</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Address</td>
<td>LONGCHAR</td>
</tr>
<tr>
<td>City</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>State/Province</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>ZIP/Postal Code</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Country/Region</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>Web Page</td>
<td>LONGCHAR</td>
</tr>
<tr>
<td>Notes</td>
<td>LONGCHAR</td>
</tr>
<tr>
<td>Attachments</td>
<td>LONGCHAR</td>
</tr>
</tbody>
</table>

Loading data from ODBC sources

`odbc load` is used to load an ODBC table into memory.

To load an ODBC table listed in the `odbc query` output, specify the table name in the `table()` option and the data source name in the `dsn()` option.

➢ Example 4

We want to load the `Employees` table from the `Northwind` data source.

```
. clear
. odbc load, table("Employees") dsn("Northwind")
```

E-mail_Address invalid name
- converted E-mail_Address to var5
State/Province invalid name
- converted State/Province to var13
ZIP/Postal_Code invalid name
- converted ZIP/Postal_Code to var14
Country/Region invalid name
- converted Country/Region to var15
. describe
Contains data
    obs:      9
    vars:     18

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>display</th>
<th>value</th>
<th>label</th>
<th>variable label</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>long</td>
<td>%12.0g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td>str17</td>
<td>%17s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last_Name</td>
<td>str14</td>
<td>%14s</td>
<td></td>
<td>Last Name</td>
<td></td>
</tr>
<tr>
<td>First_Name</td>
<td>str7</td>
<td>%9s</td>
<td></td>
<td>First Name</td>
<td></td>
</tr>
<tr>
<td>var5</td>
<td>str28</td>
<td>%28s</td>
<td></td>
<td>E-mail Address</td>
<td></td>
</tr>
<tr>
<td>Job_Title</td>
<td>str21</td>
<td>%21s</td>
<td></td>
<td>Job Title</td>
<td></td>
</tr>
<tr>
<td>Business_Phone</td>
<td>str13</td>
<td>%13s</td>
<td></td>
<td>Business Phone</td>
<td></td>
</tr>
<tr>
<td>Home_Phone</td>
<td>str13</td>
<td>%13s</td>
<td></td>
<td>Home Phone</td>
<td></td>
</tr>
<tr>
<td>Mobile_Phone</td>
<td>str1</td>
<td>%9s</td>
<td></td>
<td>Mobile Phone</td>
<td></td>
</tr>
<tr>
<td>Fax_Number</td>
<td>str13</td>
<td>%13s</td>
<td></td>
<td>Fax Number</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>strL</td>
<td>%9s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City</td>
<td>str8</td>
<td>%9s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>var13</td>
<td>str2</td>
<td>%9s</td>
<td></td>
<td>State/Province</td>
<td></td>
</tr>
<tr>
<td>var14</td>
<td>str5</td>
<td>%9s</td>
<td></td>
<td>ZIP/Postal Code</td>
<td></td>
</tr>
<tr>
<td>var15</td>
<td>str3</td>
<td>%9s</td>
<td></td>
<td>Country/Region</td>
<td></td>
</tr>
<tr>
<td>Web_Page</td>
<td>strL</td>
<td>%9s</td>
<td></td>
<td>Web Page</td>
<td></td>
</tr>
<tr>
<td>Notes</td>
<td>strL</td>
<td>%9s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attachments</td>
<td>strL</td>
<td>%9s</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sorted by:

Note: Dataset has changed since last saved.

Technical note

When Stata loads the ODBC table, data are converted from SQL data types to Stata data types. Stata does not support all SQL data types. If the column cannot be read because of incompatible data types, Stata will issue a note and skip a column. The following table lists the supported SQL data types and their corresponding Stata data types:
<table>
<thead>
<tr>
<th>SQL data type</th>
<th>Stata data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL_BIT</td>
<td>byte</td>
</tr>
<tr>
<td>SQL_TINYINT</td>
<td></td>
</tr>
<tr>
<td>SQL_SMALLINT</td>
<td>int</td>
</tr>
<tr>
<td>SQL_INTEGER</td>
<td>long</td>
</tr>
<tr>
<td>SQL_DECIMAL</td>
<td>double</td>
</tr>
<tr>
<td>SQL_NUMERIC</td>
<td></td>
</tr>
<tr>
<td>SQL_FLOAT</td>
<td>double</td>
</tr>
<tr>
<td>SQL_DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>SQL_REAL</td>
<td>double</td>
</tr>
<tr>
<td>SQL_BIGINT</td>
<td>string</td>
</tr>
<tr>
<td>SQL_CHAR</td>
<td>string</td>
</tr>
<tr>
<td>SQL_VARCHAR</td>
<td></td>
</tr>
<tr>
<td>SQL_LONGVARCHAR</td>
<td></td>
</tr>
<tr>
<td>SQL_WCHAR</td>
<td></td>
</tr>
<tr>
<td>SQL_NVARCHAR</td>
<td></td>
</tr>
<tr>
<td>SQL_WLONGVARCHAR</td>
<td></td>
</tr>
<tr>
<td>SQL_TIME</td>
<td></td>
</tr>
<tr>
<td>SQL_DATE</td>
<td></td>
</tr>
<tr>
<td>SQL_TIMESTAMP</td>
<td>double</td>
</tr>
<tr>
<td>SQL_TYPE_TIME</td>
<td></td>
</tr>
<tr>
<td>SQL_TYPE_DATE</td>
<td></td>
</tr>
<tr>
<td>SQL_TYPE_TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>SQL_BINARY</td>
<td></td>
</tr>
<tr>
<td>SQL_VARBINARY</td>
<td></td>
</tr>
<tr>
<td>SQL_LONGVARBINARY</td>
<td></td>
</tr>
</tbody>
</table>

You can also load an ODBC table generated by an SQL SELECT statement specified in the `exec()` option.

> Example 5

Suppose that, from the Northwind data source, we want a list of all the customers who have placed orders. We might use the SQL SELECT statement

```sql
SELECT DISTINCT c.ID, c.Company
FROM Customers c
INNER JOIN Orders o
  ON c.[Customer ID] = o.CustomerID
```

To load the table into Stata, we use `odbc load` with the `exec()` option.
. odbc load, exec('SELECT DISTINCT c.ID, c.Company FROM Customers c INNER JOIN Orders o ON c.ID = o.Customer ID' dsn("Northwind") clear
.describe
Contains data
  obs: 15
  vars: 2

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>display</th>
<th>value</th>
<th>label</th>
<th>variable label</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>long</td>
<td>%12.0g</td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td>Company</td>
<td>str10</td>
<td>%10s</td>
<td></td>
<td></td>
<td>Last Name</td>
</tr>
</tbody>
</table>

Sorted by:
  Note: Dataset has changed since last saved.

The extvarlist is optional. It allows you to choose which columns (variables) are to be read and to rename variables when they are read.

Example 6

Suppose that we want to load the ID column and the Last Name column from the Employees table of the Northwind data source. Moreover, we want to rename ID as id and Last Name as name.

. odbc load id=ID name="Last Name", table("Employees") dsn("Northwind") clear
.describe
Contains data
  obs: 9
  vars: 2

<table>
<thead>
<tr>
<th>variable name</th>
<th>storage</th>
<th>display</th>
<th>value</th>
<th>label</th>
<th>variable label</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>long</td>
<td>%12.0g</td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>str14</td>
<td>%14s</td>
<td></td>
<td></td>
<td>Last Name</td>
</tr>
</tbody>
</table>

Sorted by:
  Note: Dataset has changed since last saved.

The if and in qualifiers allow you to choose which rows are to be read. You can also use a WHERE clause in the SQL SELECT statement to select the rows to be read.
Example 7

Suppose that we want the information from the Order Details table, where Quantity is greater than 50. We can specify the if and in qualifiers,

\[
. odbc load if Quantity>50, table("Order Details") dsn("Northwind") clear \\
. sum Quantity
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>10</td>
<td>177.7</td>
<td>94.21966</td>
<td>87</td>
<td>300</td>
</tr>
</tbody>
</table>

or we can issue the SQL SELECT statement directly:

\[
. odbc load, exec("SELECT * FROM [Order Details] WHERE Quantity>50") > dsn("Northwind") clear \\
. sum Quantity
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>10</td>
<td>177.7</td>
<td>94.21966</td>
<td>87</td>
<td>300</td>
</tr>
</tbody>
</table>

Example 8

To use odbc insert, you must have an SQL table already created in your data source. If you do not, you can use odbc exec to create a table in your data source. For example, one might create a table in an Oracle database with the SQL command below:

\[
#delimit ; \\
local cols '" ID NUMBER(5,0), \\
      PAY NUMBER(8,2), \\
      TITLE NVARCHAR2(18), \\
      LOCATION VARCHAR(40)"; ; \\
#delimit cr \\
odbc exec('"CREATE TABLE JOB_TYPES ('cols');"'), dsn(oracle_dsn) // user(username) password(password)
\]

You must create a table using the correct data type for each table column for your data to transfer correctly. Note that the SQL syntax to create a table differs across data sources, as do column data types.

Reference


Also see

[D] export — Overview of exporting data from Stata
[D] import — Overview of importing data into Stata