

TITANIUM Data Extraction Tool (TDEX) Version 6.2

Savitar Corporation

Legal Notices

Copyright © 2004, 2005 Savitar Corporation.

These manuals are provided for the use of customers and customer employees. The entire contents have been copyrighted by Savitar Corporation. Copying, duplicating, selling, or other distribution of the software is a violation of the copyright laws. Reproduction by any means of the software or documentation is prohibited except as permitted in a written agreement with Savitar Corporation.

TITANIUM, GURU, and all other Savitar products including legacy **mdbs** products are copyrighted computer software, and all rights, including intellectual property rights, are reserved by Savitar Corporation. Use of software requires an End User License. Please refer to the End User License Agreement for the corresponding terms and conditions.

TITANIUM is protected by U.S. Patent Nos. 5,611,076 and 5,713,014 in addition to foreign patents and applications pending.

Limitation on Warranties and Liability

The documentation in these manuals is provided for reference purposes only. Savitar Corporation has taken great pains to ensure that the manuals reasonably conform to the software. However, Savitar Corporation reserves the right to make changes in specifications and other information in these publications without prior notice.

Savitar Corporation makes no warranties, either express or implied, respecting the software and documentation or its quality, performance, merchantability, or fitness for any particular purpose. Savitar Corporation software and documentation is licensed for use "as is."

Savitar Corporation shall not be liable to the user or any other person or entity with respect to any liability, loss or damage caused by, or alleged to be caused by, the software and documentation or its use, whether direct, incidental, or consequential, according to the terms of the license agreement.

Trademarks

TITANIUM[®] and GURU[®] are a registered trademarks of Micro Data Base Systems. MDBS IV[®], MDBS III[®], KnowledgeMan[®] and Object/1[®] are also registered trademarks of Micro Data Base Systems.

Other brands and product names may be trademarks or registered trademarks of their respective holders.

TITANIUM Data Extraction Tool (TDEX)

This document describes the TITANIUM Data Extraction Tool (TDEX) version 6.2.

System Requirements

The TITANIUM Data Extraction Tool (TDEX) version runs on a Microsoft Windows 32-bit (Windows 2000 or Windows XP) database server machine. TDEX 6.2 requires the TITANIUM Database Engine shared memory server (TDESRV) version 6.1e or greater to be already installed on the server.

TDEX is licensed software; a license agreement must be executed and a separate license must be purchased for each TDEX installation. Please see the TDEX license agreement for details.

Installation

The TITANIUM Data Extraction Tool (TDEX) version is shipped on a compact disc (CD) containing the TDEX executable (TDEX.EXE) and with example command and support files. No installation procedure is required; simply copy the files into a directory on the target machine .

Other Required Files

The TITANIUM Data Extraction Tool (TDEX) is based on the TITANIUM SQL Access Manager technology and requires a Table Definition File (.TDF) file containing definitions of tables contained in the TITANIUM database as well as one or more SQL command files (.SQL) to query data from the TDF-defined tables. Examples of these files are provided with the TDEX utility; a full description is contained in the TITANIUM SQL Access Manager manual. A TDF is a text file that may be created and edited directly in any text editor, or generated with utilities such as TDDL2TDF, available separately.

Operation: Starting the Database Server

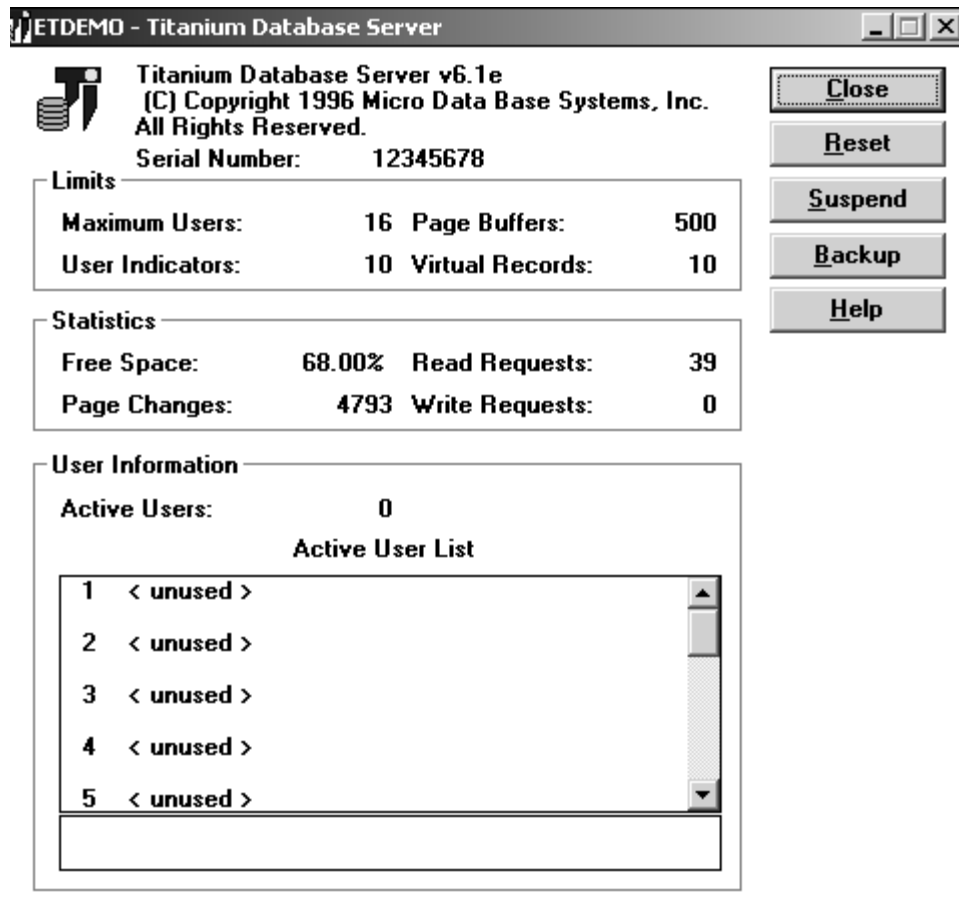
The TITANIUM Data Extraction Tool (TDEX) tool is a command line utility that extracts data from a running TITANIUM database and exports it to an XML data format.

Operation requires that the TITANIUM shared memory database server (TDESRV.EXE) be running on the database of interest. To start the database server, enter the TDESRV command in a command prompt window with default directory containing the database files. For example, a database named ETDEMO may be started with:

```
TDESRV -NB500 -DETDEMO
```

The -NB flag sets the number of page buffers (memory cache size) for the database; a larger number will generally improve performance if physical memory is available. The example shown sets a buffer of 500 pages, resulting in a 1 megabyte cache assuming a database page size of 2k.

When the database server is started, the following screen will appear on the database server:



Operation: Executing TDEX

Once the database server is running, execute TDEX from the command line using the command line options to specify the database, TDF file, and SQL command file. The generated XML is sent to the standard output. To save the generated XML to a file, use the -fbo option. For example, this command uses the provided sample files to generate XML output.:

```
tdex -detdemo.tdf -fboetdemo.xml -uuser -ppass -xml etdemo.sql
```

In the above example, the -D flag specifies the ETDEMO.TDF table definition file as the source of the database and table definitions. The username “user” is specified with the -U flag; the password “pass” is specified with the -P flag. The -FBO option directs the output to the file ETDEMO.XML. The -XML flag specifies XML output, though that is the default format for TDEX and is not required. The SQL command file, ETDEMO.SQL, extracts all data from the tables listed in the TDF with standard SQL commands as shown here:

```
select * from EmployeesByName;  
select * from EmployeesByIDNumber;  
select * from Basic_Employee_Info;
```

Progress information is sent to the command window listing the number of rows processed for each table:

```
<54 rows fetched with 2 columns into EmployeesByName elements>
```

```
<54 rows fetched with 2 columns into EmployeesByIDNumber elements>
```

```
<253 rows fetched with 6 columns into Basic_Employee_Info elements>
```

The command prompt reappears when the extraction process is finished. Note that a QUIT; command is required at the end of the SQL file to terminate the session.

XML Output

The output XML contains a <ROOT> element. Each row of the table is listed hierarchically under the root element with a tag containing the table name. Each field of the row is contained within tags containing the field name. The given example produces XML output as shown below:

```
<?xml version="1.0"?>  
<ROOT>  
  <EmployeesByName>  
    <EMPLOYEE_EMPLOYEEENAME>Alps, Pat</EMPLOYEE_EMPLOYEEENAME>  
    <EMPLOYEE_EMPLOYEEIDNUMBER>70</EMPLOYEE_EMPLOYEEIDNUMBER>  
  </EmployeesByName>  
  <EmployeesByName>  
    <EMPLOYEE_EMPLOYEEENAME>Arro, Bill</EMPLOYEE_EMPLOYEEENAME>  
    <EMPLOYEE_EMPLOYEEIDNUMBER>66</EMPLOYEE_EMPLOYEEIDNUMBER>  
  </EmployeesByName>  
  . . .  
</ROOT>
```

Modifying the Database

TDEX is a read-only tool and does not support modification of the database.

SQL-92 and ODBC 3.0 Compliance

The TITANIUM Data Extraction Tool (TDEX) supports the SQL-92 Entry Level SQL grammar as defined in the ANSI SQL-92 standard and the Microsoft ODBC version 3.0 documentation. SQL SELECT syntax may be used to select a subset of desired data from a TITANIUM database.

SQL Built-In Functions

The following TITANIUM SQL built-in function is of potential use in TDEX SQL command files to establish surrogate keys for use in exporting data to relational databases:

DBKEY(*column*)

(TITANIUM SQL extension) This function returns the TITANIUM database key of the TITANIUM record from which the value of *column* is derived. The database key may be passed to a user-defined SQL function implemented as a TITANIUM C Stored Procedure. Each TITANIUM database key is a long unsigned integer. Only columns defined in the current TDF or for the current database are recognized by this function. It does not recognize complex expressions.

TDEX Reference

The following section lists TDEX command line options and interactive commands.

Running TDEX

To start a TDEX session, enter a command in the following format on the operating system command line:

```
TDEX [optional arguments] [optional command files]
```

For example, the following command starts an interactive TDEX session against tables defined by the ETDEMO table definition file:

```
TDEX -detdemo.tdf
```

TDEX prompts for the name of the database, a schema name, a valid database user name, and a valid password for the user name. The optional command line flags (e.g. `-ddatabase -snschemaname -username -ppassword`) can be used to specify this information to bypass the TDEX prompts. If *database* is a table definition file that includes a user name and password, TDEX does not prompt for such information.

Once TDEX has successfully opened the database, it checks for the presence of one or more command files. If command files are specified on the command line, TDEX runs in batch mode, processing the contents of the command files as SQL commands against the database.

If no command file is specified, TDEX runs in interactive mode. The `?>` prompt appears, and TDEX is ready to accept commands. All commands must be terminated with a semicolon (`;`). After TDEX responds to a command, the `?>` prompt appears again to show that TDEX is ready to accept the next command

For more information, see “Executing SQL Commands” on page : 8.

Quitting TDEX

End an interactive TDEX session by entering:

`QUIT`

This command automatically closes the database and returns control to the operating system.

TDEX Command Line Arguments

The following arguments can be included on the TDEX command line to specify the behavior of the session. End users must supply their own words and letters for those shown in *italics* for each database they open; the angle brackets (`<>`) beside those words must not be typed as part of the argument.

Argument	Description
<code>-@<filename></code>	Specifies a file containing a list of startup arguments, each on its own line.
<code>-d <filename></code>	Specifies the database to use with TDEX. May be a TITANIUM database (<code>*.DB</code>), table definition file (<code>*.TDF</code>), or compiled table definition file (<code>*.TDC</code>). If this argument is omitted from the command line, TDEX prompts the user for this information.
<code>-e</code>	Echoes SQL command to output.
<code>-fbo<filename></code>	Send output to <code><filename></code> instead of standard output.
<code>-h</code>	Displays a summary of TDEX command line options.
<code>-htm</code>	Causes command results to be output as tables in HTML format.
<code>-k</code>	Toggles query optimizer. Default is ON.
<code>-log<filename></code>	Specifies a file for trace logging.
<code>-man</code>	Specifies manual transactions. Default is OFF.
<code>-mc<n></code>	Specifies maximum width of returned columns. Default is 80.
<code>-mr<n></code>	Specifies maximum number of returned rows. Default is 0 (no limit).
<code>-muo</code>	Suppresses (mutes) the display of the initial banner message for TDEX.
<code>-nb<n></code>	Specifies the number of page buffers to use. Default is 15, 30, or 60 depending on page size (single-user only).
<code>-ns<server></code>	Specifies a database server name.

Argument	Description
-nx	Does not execute queries. When this mode is active, queries are compiled and prepared, but no data is retrieved.
-p<password>	Specifies a valid user password for the database. If this argument is omitted for a table definition file database, TDEX attempts to extract a password from the table definition file. If no password is specified on the command line or in a table definition file, TDEX prompts for a valid password before providing access to the database.
-r<n>	Specifies number of retries on lock. Default is 0.
-sn<schema>	Specifies database schema name. Default is database base name.
-t<n>	Reties interval on lock (in seconds). Default is no retry.
-tr	Toggles column display truncation. Default is OFF.
-u<username>	Specifies a valid user name for the database. If this argument is omitted for a table definition file database, TDEX attempts to extract a user name from the table definition file. If no user name is specified on the command line or in a table definition file, TDEX prompts for a valid user name before providing access to the database.
-v	Specifies verbose output, similar to ODBC trace logging.
-wd<path>	Specifies working directory, as used by ODBC.
-xml	Specifies XML format output; this is the default output format for TDEX and so the flag -xml is redundant.
-xav	XML output in Attribute Value form, such as <ROW COL1="val1" COL2="val2" />
-xhr	XML increased human readability with extra spacing between records.
-xon	XML omit NULL values (this is on by default if -xav Attribute Value form specified)
-xrn	XML row name (default: table name, first table if multi-table query)
-xzn	XML convert zero length strings to NULL. To omit blank records entirely from the output, specify both -xzn and -xav.

TDEX Commands

TDEX supports several commands in addition to SQL syntax. Many of these commands can be used to determine the configuration of the TDEX session. Many duplicate the effects of command line arguments. Some commands accept an optional parameter, indicated below by square brackets ([?]). Executing such commands without a parameter typically sets a default value.

The --: (dash dash colon) prefix before certain commands distinguishes it from SQL syntax. The -- characters at the beginning of the command corresponds to comment indicators in SQL syntax, allowing TDEX commands to be embedded in SQL command files without generating errors. Any TDEX command may be preceded by the --: prefix (e.g., --: PRINT LOG), for use in SQL batch file processing. The rest of the line, after the TDEX command, is treated as a comment.

Command	Effect
HELP	Provides online descriptions of TDEX command usage.

Command	Effect
PRINT [VERBOSE ECHO NOEXEC LOG MR MC MANUAL TRUNCATE]	Displays the status of the specified TDEX configuration option. If no configuration option is specified, the statuses of all options are listed.
QUIT	Ends the TDEX session.
--:QUIT	Ends the TDEX session.
--:ECHO [T F]	Echoes command line when reading from script.
--:LOG [<filename>]	Specifies a file for trace logging (same as the -log command line argument). If no file is specified, logging is turned off.
--:MC [<num>]	Sets the maximum width of returned columns (same as the -mc command line argument). Default is 80.
--:MR [<num>]	Sets the maximum number of returned rows (same as the -mr command line argument). Default is 0, indicating no limit.
--:MANUAL [T F]	Turns on/off manual transactions (similar to man command line argument).
--:MSG	Prints a message (for example, when executing a script).
--:NOEXEC [T F]	Turns on/off query execution status (similar to the -nx command line argument). If query execution is turned off, queries are compiled and prepared, but no data is retrieved. If no parameter is specified, the status toggles.
--:READ <filename>	Reads and executes the specified SQL command file in batch mode. Such files may themselves contain --:READ commands.
--:VERBOSE [T F]	Turns on/off verbose output (similar to the -v command line argument). If no parameter is specified, the status toggles.

Executing SQL Commands

TDEX understands the entire set of SQL commands in SQL Access Manager. In interactive mode, TDEX awaits each new command with the question mark-greater than (?>) prompt. Each SQL command must be terminated with a semicolon (;). Commands that are not terminated are expected to continue on the next line. TDEX awaits the remainder of a non-terminated command with the space-greater than continuation prompt (>). Once the command is terminated with a semicolon and executed, TDEX again displays the ?> prompt.

```
?> select table_name from information_schema.tables
> ;
TABLE_NAME
-----
COLUMN_PRIVILEGES
COLUMNS
FOREIGN_KEYS
. . .
```

```

USERS
GLOBALS
<18 rows fetched with 1 column>
?>

```

Binary and BLOB Limitations

Note: TDEBinary and BLOB data items have only limited support in TDEX. TDEX displays the contents of such data items as raw data in hexadecimal format, with limited length for BLOBs.

Sample Table Definition File (ETDEMO.TDF)

This Appendix provides a listing of the sample Table Definition File that can be used with the TITANIUM Data Extraction Tool (TDEX): ETDEMO.TDF.

```

-- SQL Access Manager Table Definition File
-- Provides access to the ETDEMO database

DATABASE employee_demo
LOGICAL 'ETDEMO'
USER 'user'
PASS 'pass'

-- Table definition section

VIEW Basic_Employee_Info
REMARKS 'Employee name, ID, task, & dept.'
      Employee.EmployeeName
      Employee.EmployeeIDNumber
      Task.TaskName
      Task.TaskIDNumber
      Department.DepartmentName
      Department.DepartmentIDNumber
FROM DepartmentsByName, Staffed, Assigned

-- Do not make a table with two or more system sets similar to the following.
-- It generates the cross product of all the fields.
-- In this case, there are 7 departments and 54 employees which
-- gives you a total of 378 rows in the results of a query on this table.
-- See the Employee_Departments table below for a correct form of this table.

```

TITANIUM Data Extraction Tool (TDEX)

```
VIEW Bad_Employee_Info
REMARKS 'Shows "bad data" in a cross-product'
      Employee.EmployeeName
      Employee.EmployeeIDNumber
      Department.DepartmentName
      Department.DepartmentIDNumber
FROM DepartmentsByName, EmployeesByName

-- This table uses the EmployeesByName or EmployeesByIDNumber system set,
-- depending on whether or not a WHERE clause condition specifies "WHERE
-- EmployeeName ..." or "WHERE EmployeeIDNumber ...". If no WHERE clause is
-- specified, this table uses the first FROM clause, which is sorted by
-- EmployeeName.

BASE TABLE Employee_Info_Byemps
REMARKS 'Sorted by EmployeeName'
      Employee.EmployeeName
      Employee.EmployeeIDNumber
      EmployeeBiography.DateOfHire
      SalariedEmployeeDetails.Salary
      HourlyEmployeeDetails.HourlyRate
      HourlyEmployeeDetails.WorkShift [1]      AS      Start_Time
      HourlyEmployeeDetails.WorkShift [2]      AS      Stop_Time
      CommissionedEmployeeDetails.Basesalary
      CommissionedEmployeeDetails.CommissionPercentage
FROM EmployeesByName, Has, Earns
FROM EmployeesByIDNumber, Has, Earns

-- This table and the next can be used for simple queries and
-- for JOINing with other tables if necessary.

VIEW Employee_Tasks
      Employee.EmployeeName
      Employee.EmployeeIDNumber
      Task.TaskName
      Task.TaskIDNumber
FROM TasksByName, Assigned

VIEW Employee_Departments
      Employee.EmployeeName
      Employee.EmployeeIDNumber
      Department.DepartmentName
      Department.DepartmentIDNumber
FROM DepartmentsByName, Staffed

-- Use this table if complete employee information including payroll is
-- desired.

/* Because this table joins several entities in the database it can only be
** used to do updates and deletions in a query. It cannot be opened with
** tools like Microsoft Access to insert, update, or delete a row from
** the table. The comments can be removed to see how update trigger works.
*/

BASE TABLE Employee_Biography
--      Update Trigger "UpTrig"
      Employee.EmployeeName
      Employee.EmployeeIDNumber
```

```

        EmployeeBiography.DateOfHire
        EmployeeBiography.StreetAddress [1]    AS    Address1
REMARKS 'Street address'
        EmployeeBiography.StreetAddress [2]    AS    Address2
REMARKS 'Apt./Suite info'
        EmployeeBiography.City
        EmployeeBiography.State
        EmployeeBiography.ZipCode
        EmployeeBiography.OtherInformation
FROM EmployeesByName, Has

-- Use this table if employee, department, hiredate and payroll information
-- is desired.

VIEW Employee_Depts_Payroll
    Employee.EmployeeName
    Employee.EmployeeIDNumber
    EmployeeBiography.DateOfHire
    Department.DepartmentName
    Department.DepartmentIDNumber
    SalariedEmployeeDetails.Salary
    HourlyEmployeeDetails.HourlyRate
    HourlyEmployeeDetails.WorkShift [1]    AS    Start_Time
    HourlyEmployeeDetails.WorkShift [2]    AS    Stop_Time
    CommissionedEmployeeDetails.BaseSalary
    CommissionedEmployeeDetails.CommissionPercentage
FROM DepartmentsByName, Staffed, Has, Earns

-- To get employee, department, task, hiredate, and pay information
-- in one report, JOIN the Employee_Tasks table with the
-- Employee_Depts_Payroll table in the SQL query.
-- Remember, Employee has a many-to-many relationship with Tasks.

/* This base table can be use in a front end tool such as Microsoft Access
* to update, delete, or insert a row without writing a query. IsrtTrig can
* also be used to create an entry in the table that has a 1:1 relationship
* with Employee. Then create an update query that fills in the correct
* information using Employee_Biography table.
*/
BASE TABLE EmployeesByName
DATABASE employee_demo
    Employee.EmployeeName    AS    Employee_EmployeeName
    Employee.EmployeeIDNumber AS    Employee_EmployeeIDnumber
FROM EmployeesByName

/* This table get the same information as above, but is ordered by ID number
** rather than by name. The delete trigger is a csp that does not allow
** the employee with ID number 66 to be deleted. To see how it work remove
** the comment from the line.
*/

BASE TABLE EmployeesByIDNumber
-- Delete Trigger "DeleteTrigger"
    Employee.EmployeeName    AS    Employee_EmployeeName
    Employee.EmployeeIDNumber AS    Employee_EmployeeIDnumber
FROM EmployeesByIDNumber
-- end etdemo.tdf

```

Example ANSI SQL Queries

The following are examples of ANSI SQL queries supported by SQL Access Manager. This list represents commonly-used SQL queries. It is not a comprehensive list of allowed queries. The examples are based on the following sample Table Definition File:

```
DATABASE db1
LOGICAL 'ITEMS'

BASE TABLE table1 REMARKS 'example table'
    re1.item1 AS col1
    re1.item2 AS col2
    re1.item3 AS col3
    re1.item4 AS col4
FROM set1
```

Commonly-Used SQL Queries

```
-- simple SELECT
SELECT col1
FROM table1

-- SELECT with wildcard (all columns)
SELECT *
FROM table1

-- SELECT with WHERE expression
SELECT *
FROM table1
WHERE col1 = 'text'           -- column 1 equal to literal string 'text'

SELECT *
FROM table1
WHERE col1 <> 'text'         -- column 1 not equal to literal string
                             -- 'text'

-- SELECT with WHERE and AND
SELECT col2, col3
FROM table1
WHERE col3 > '06/30/1985' AND col3 < '12/31/1986'

-- SELECT with WHERE and BETWEEN
SELECT col2, col3
FROM table1
WHERE col2 BETWEEN 120 AND 400

-- SELECT with WHERE and LIKE
SELECT col1, col3
FROM table1
WHERE col1 LIKE '%text'     -- any column one ending with 'text'
```

```
SELECT *
FROM table1
WHERE col1 LIKE '[ABC]%'
-- any column one beginning with A, B, or C

SELECT *
FROM table1
WHERE col1 LIKE 'A_'
-- any column one consisting of A followed by 1
-- character

-- SELECT using IN
SELECT *
FROM table1
WHERE col1
IN ('text1', 'text2', 'text3')
-- column one equal to any expression in list

-- SELECT using DISTINCT
SELECT DISTINCT col1
FROM table1
-- unique values of col1

-- SELECT using SUBSTRING function
-- extract string of 3 characters starting at position 4
SELECT col2, SUBSTRING(col2, 4, 3)
FROM table1

-- SELECT using LTRIM function
-- trim leading blanks from column one
SELECT LTRIM(col1)
FROM table1

-- SELECT using RTRIM function
-- trim trailing blanks from column one
SELECT RTRIM(col1)
FROM table1

-- SELECT using aggregate functions SUM, AVG, MIN, MAX, COUNT:
SELECT SUM(col2)
FROM table1
-- sum of all column two

SELECT AVG(col2)
FROM table1
-- average of column two

SELECT MAX(col2)
FROM table1
-- maximum of column two

SELECT MIN(col2)
FROM table1
-- minimum of column two

SELECT COUNT(col2)
FROM table1
-- count of rows in table1

SELECT COUNT(DISTINCT col2)
FROM table1
-- count of distinct rows in table1

-- SELECT using concatenation operator
-- make single column of form col1,col2
SELECT (col1 || ', ' || col2 )
```

TITANIUM Data Extraction Tool (TDEX)

```
FROM table1

-- SELECT using CONVERT function
SELECT CONVERT (col2, INTEGER)           -- column two as integer
FROM table1

SELECT CONVERT (col2, CHARACTER(20))     -- column two as character
length 20
FROM table1

-- SELECT using ORDER BY
SELECT col1, col2, col3
FROM table1
ORDER BY col2                            -- order by ascending values of column two

SELECT col1, col2, col3
FROM table1
ORDER BY col2 DESC                       -- order by descending values of column two

SELECT col1, col2, col3
FROM table1
ORDER BY col2 DESC, col3, col1          -- order by descending values of column two
                                         -- then ascending values of column three

-- SELECT using GROUP BY
SELECT col1, col2, col3
FROM table1
GROUP BY col2    -- group together rows with same values of column two
```