Mimer

Core SQL Feature Summary

The following table lists all features included in Core SQL.

It also indicates if Mimer SQL supports a specific feature.

Feature	Feature Name	Mimer SQL	Core SQL
E011	**Numeric data types**	√	√
E011–01	INTEGER and SMALLINT data types (including all spellings)	√	√
E011–02	REAL, DOUBLE PRECISON, and FLOAT data types	√	√
E011–03	DECIMAL and NUMERIC data types	√	√
E011–04	Arithmetic operators	√	√
E011–05	Numeric comparison	√	√
E011–06	Implicit casting among the numeric data types	√	√
E021	**Character string types**	√	√
E021–01	CHARACTER data type (including all its spellings)	√	√
E021–02	CHARACTER VARYING data type (including all its spellings)	√	√
E021–03	Character literals	√	√
E021–04	CHARACTER_LENGTH function	√	√
E021–05	OCTET_LENGTH function	√	√
E021–06	SUBSTRING function	√	√
E021–07	Character concatenation	√	√
E021–08	UPPER and LOWER functions	√	√
E021–09	TRIM function	√	√
E021–10	Implicit casting among the fixed-length and variable-length character string types	√	√
E021–11	POSITION function	√	√
E021–12	Character comparison	√	√
E031	**Identifiers**	√	√
E031–01	Delimited identifiers	√	√
E031–02	Lower case identifiers	√	√
E031–03	Trailing underscore	√	√
E051	**Basic query specification**	√	√
E051–01	SELECT DISTINCT	√	√
E051–02	GROUP BY clause	√	√
E051–04	GROUP BY can contain columns not in <select-list>	√	√
E051–05	Select list items can be renamed	√	√
E051–06	HAVING clause	√	√
E051–07	Qualified * in select list	√	√
E051–08	Correlation names in the FROM clause	√	√
E051–09	Rename columns in the FROM clause	√	√
E061	**Basic predicates and search conditions**	√	√
E061–01	Comparison predicate	√	√
E061–02	BETWEEN predicate	√	√
E061–03	IN predicate with list of values	√	√
E061–04	LIKE predicate	√	√
E061–05	LIKE predicate: ESCAPE clause	√	√
E061–06	NULL predicate	√	√
E061–07	Quantified comparison predicate	√	√
E061–08	EXISTS predicate	√	√
E061–09	Subqueries in comparison predicate	√	√
E061–11	Subqueries in IN predicate	√	√
E061–12	Subqueries in quantified comparison predicate	√	√

E061–13	Correlated subqueries	√	√
E061–14	Search condition	√	√
E071	**Basic query expressions**	√	√
E071–01	UNION DISTINCT table operator	√	√
E071–02	UNION ALL table operator	√	√
E071–03	EXCEPT DISTINCT table operator	√	√
E071–05	Columns combined via table operators need not have exactly the same data type	√	√
E071–06	Table operators in subqueries	√)	√
E081	**Basic Privileges**	√	√
E081–01	SELECT privilege at the table level	√	√
E081–02	DELETE privilege	√	√
E081–03	INSERT privilege at the table level	√	√
E081–04	UPDATE privilege at the table level	√	√
E081–05	UPDATE privilege at the column level	√	√
E081–06	REFERENCES privilege at the table level	√	√
E081–07	REFERENCES privilege at the column level	√	√
E081–08	WITH GRANT OPTION	√	√
E081–09	USAGE privilege	√	√
E081–10	EXECUTE privilege	√	√
E091	**Set functions**	√	√
E091–01	AVG	√	√
E091–02	COUNT	√	√
E091–03	MAX	√	√
E091–04	MIN	√	√
E091–05	SUM	√	√
E091–06	ALL quantifier	√	√
E091–07	DISTINCT quantifier	√	√
E101	**Basic data manipulation**	√	√
E101–01	INSERT statement	√	√
E101–03	Searched UPDATE statement	√	√
E101–04	Searched DELETE statement	√	√
E111	**Single row SELECT statement**	√	√
E121	**Basic cursor support**	√	√
E121–01	DECLARE CURSOR	√	√
E121–02	ORDER BY columns need not be in select list	√	√
E121–03	Value expressions in ORDER BY clause	√	√
E121–04	OPEN statement	√	√
E121–06	Positioned UPDATE statement	√	√
E121–07	Positioned DELETE statement	√	√
E121–08	CLOSE statement	√	√
E121–10	FETCH statement: implicit NEXT	√	√
E121–17	WITH HOLD cursors	√	√
E131	**Null value support (nulls in lieu of values)**	√	√
E141	**Basic integrity constraints**	√	√
E141–01	NOT NULL constraints	√	√
E141–02	UNIQUE constraints of NOT NULL columns	√	√
E141–03	PRIMARY KEY constraints	√	√
E141–04	Basic FOREIGN KEY constraint with the NO ACTION default for both referential delete action and referential update action	√	√
E141–06	CHECK constraints	√	√
E141–07	Column defaults	√	√
E141–08	NOT NULL inferred on PRIMARY KEY	√	√
E141–10	Names in a foreign key can be specified in any order	√	√
E151	**Transaction support**	√	√
E151–01	COMMIT statement	√	√
E151–02	ROLLBACK statement	√	√
E152	**Basic SET TRANSACTION statement**	√	√

E152–01	SET TRANSACTION statement: ISOLATION LEVEL SERIALIZABLE clause	√	√
E152–02	SET TRANSACTION statement: READ ONLY and READ WRITE clauses	√	√
E153	**Updatable queries with subqueries**	√	√
E161	**SQL comments using leading double minus**	√	√
E171	**SQLSTATE support**	√	√
E182	**Host language Binding (previously "Module Language")** NOTE-1)	√	√
F021	**Basic information schema**	√	√
F021–01	COLUMNS view	√	√
F021–02	TABLES view	√	√
F021–03	VIEWS view	√	√
F021–04	TABLE_CONSTRAINTS view	√	√
F021–05	REFERENTIAL_CONSTRAINTS view	√	√
F021–06	CHECK_CONSTRAINTS view	√	√
F031	**Basic schema manipulation**	√	√
F031–01	CREATE TABLE statement to create persistent base tables	√	√
F031–02	CREATE VIEW statement	√	√
F031–03	GRANT statement	√	√
F031–04	ALTER TABLE statement: ADD COLUMN clause	√	√
F031–13	DROP TABLE statement: RESTRICT clause	√	√
F031–16	DROP VIEW statement: RESTRICT clause	√	√
F031–19	REVOKE statement: RESTRICT clause	√	√
F041	**Basic joined table**	√	√
F041–01	Inner join (but not necessarily the INNER keyword)	√	√
F041–02	INNER keyword	√	√
F041–03	LEFT OUTER JOIN	√	√
F041–04	RIGHT OUTER JOIN	√	√
F041–05	Outer joins can be nested	√	√
F041–07	The inner table in a left or right outer join can also be used in an inner join	√	√
F041–08	All comparison operators are supported (rather than just =)	√	√
F051	**Basic date and time**	√	√
F051–01	DATE data type (including support of DATE literal)	√	√
F051–02	TIME data type (including support of TIME literal) with fractional seconds precision of at least 0	√	√
F051–03	TIMESTAMP data type (including support of TIMESTAMP literal) with fractional seconds precision of at least 0 and 6	√	√
F051–04	Comparison predicate on DATE, TIME, and TIMESTAMP data types	√	√
F051–05	Explicit CAST between datetime types and character string types	√	√
F051–06	CURRENT_DATE	√	√
F051–07	LOCALTIME	√	√
F051–08	LOCALTIMESTAMP	√	√
F081	**UNION and EXCEPT in views**	√	√
F131	**Grouped operations**	√	√
F131–01	WHERE, GROUP BY, and HAVING clauses supported in queries with grouped views	√	√
F131–02	Multiple tables supported in queries with grouped views	√	√
F131–03	Set functions supported in queries with grouped views	√	√
F131–04	Subqueries with GROUP BY and HAVING clauses and grouped views	√	√
F131–05	Single row SELECT with GROUP BY and HAVING clauses and grouped views	√	√
F181	**Multiple module support** NOTE-2)	√	√
F201	**CAST function** NOTE-3)	√	√
F221	**Explicit defaults** NOTE-4)	√	√
F261	**CASE expression**	√	√
F261–01	Simple CASE	√	√
F261–02	Searched CASE	√	√
F261–03	NULLIF	√	√
F261–04	COALESCE	√	√
F311	**Schema definition statement**	√	√
F311–01	CREATE SCHEMA	√	√

F311–02	CREATE TABLE for persistent base tables	√	√
F311–03	CREATE VIEW	√	√
F311–04	CREATE VIEW: WITH CHECK OPTION	√	√
F311–05	GRANT statement	√	√
F471	**Scalar subquery values**	√	√
F481	**Expanded NULL predicate**	√	√
F501	**Features and conformance views**	√	√
F501–01	SQL_FEATURES view	√	√
F501–02	SQL_SIZING view	√	√
F501–03	SQL_LANGUAGES view	√	√
F812	**Basic flagging** NOTE-5)	√	√
S011	**Distinct data types**	√	√
S011–01	USER_DEFINED_TYPES view	√	√
T321	**Basic SQL-invoked routines** NOTE-6)	√	√
T321–01	User-defined functions with no overloading	√	√
T321–02	User-defined stored procedures with no overloading	√	√
T321–03	Function invocation	√	√
T321–04	CALL statement	√	√
T321–05	RETURN statement	√	√
T321–06	ROUTINES view	√	√
T321–07	PARAMETERS view	√	√
T631	**IN predicate with one list element**	√	√

NOTE-1) An SQL-implementation is required to supply at least one binding to a standard host language using either module language, embedded SQL, or both. This can be through the support of any of the features B011 through B117.

NOTE-2) The ability to associate multiple host compilation units with a single SQL-session at one time.

NOTE-3) This means the support of CAST, where relevant, among all supported data types.

NOTE-4) Including its use in UPDATE and INSERT statements.

NOTE-5) This form of flagging identifies vendor extensions and other non-standard SQL by checking syntax only without requiring access to the catalog information.

NOTE-6) "Routine" is the collective term for functions, methods, and procedures. This feature requires a conforming SQL-implementation to support both user-defined functions and user-defined procedures. An SQL-implementation that conforms to Core SQL shall support at least one language for writing routines; that language may be SQL. If the language is SQL, then the basic specification capability in Core SQL is the ability to specify a one-statement routine. Support for overloaded functions and procedures is not part of Core SQL