Table 8

SQL Data Manipulation Language (DML) Expanded Construct Set

DML Constructs	Justification
Selecting data	The means by which to extract data from a database management system.
• Joins: outer and semi	In addition to the joins in the minimum construct set, outer and semi joins are necessary for advanced tasks and are generally more confusing in terms of appropriate application. Outer joins include tuples that do not have a match during a join. For example, they can be instrumental in database merging and extracting, transforming, and loading (ETL) operations that require all data to be migrated. Semi joins provide significant performance gains when data are transmitted and joined over a network. Examples include distributed and federated databases.
• Correlated subqueries and [NOT] EXISTS	The result of a correlated subquery is dependent upon an element within the current tuple in the outer query. For instance, to determine the maximum weight recorded for *each* patient, the subquery locates the maximum weight for the current patient in the outer query. Many questions in healthcare rely on applying subquestions to elements within sets, substantiating the inclusion of this construct in this set. Existence is essentially a Boolean response to the correlated subquery problem; that is, values exist or they do not.
• Views	While the minimum construct set promotes basic knowledge, expanded instruction should place greater emphasis on view construction to assist in complex query environments and optimization.
Combining sets: UNION [ALL], INTERSECT [ALL], and EXCEPT [ALL]	When working with multiple sets (e.g., query results or relations), it is sometimes necessary to combine or limit given elements in another. For instance, a researcher might be interested in patients with certain diseases who do not take medications in a given class. Assuming that diseases and medications are assigned to encounters and are not associated with one another, it would be impossible to join on the two in a meaningful way (other than a Cartesian join). Thus, query A would produce the set of patients meeting the disease requirement, and query B would produce the inverse medication constraint (i.e., NOT IN becomes IN to collect those in the set for removal). Patients in set A who also exist in set B are removed (EXCEPT), producing the desired patient set. This scenario is just one of many healthcare examples in which these set combination operators are necessary. However, because set combination requires a detailed understanding of set theory and basic query design, it is considered an advanced topic.