1. Statements, expressions, queries
   1. URIs
      1. <https://docs.oracle.com/cd/B28359_01/server.111/b28286/expressions001.htm#SQLRF52045>
      2. <http://beginner-sql-tutorial.com/sql-commands.htm>
      3. <http://www.informit.com/articles/article.aspx?p=29661> (there are 6 pages that comprise this source; they are found by clicking “Next >” which is next to “Page 1 of 6” near the bottom of the page)

An expression in SQL is a group of at least one value literal or SQL function call that, when used, returns a value of some sort (number, boolean, etc.) that can be used directly within an SQL statement. SQL statements are a group of at least one expression and reserved SQL keywords (using the word ‘keywords’ in the broadest sense) that synergize to create the smallest unit of executable code. An SQL query is a statement that performs a SELECT, UPDATE, INSERT, or DELETE on some data/rows of data; it is pretty much a “question” that is asked of the database. An expression, when combined with some more SQL keywords, can form a statement, which can be a query if it meets the above requirement.

The terms ‘expression’ and ‘query’ are very commonly used, and ‘statement’ is most often preceded by ‘select’ when discussing such elements of SQL. This is predominantly because the select statement is the most commonly-used statement throughout all of SQL. Thus, the select statement is the most commonly-used query.

* 1. One of the most important ideas to keep in mind when discussing the quantum pieces of SQL is that expressions and statements are unchangeably related yet are still very different concepts. Expressions are the bricks of a symbolic wall that is the SQL statement, and SQL keywords are the mortar. The bricks (expressions) must be properly put together with each other using mortar (keywords) in order to have a sturdy and sound wall (statement).

One other important concept to recognize is that select statements are used often enough such that most people consider the word ‘query’ to be synonymous with ‘select statement.’ When looking for support online (when one inevitably encounters an error when using SQL), if one does not know this bit of vernacular, one would have to spend a few more minutes trying to figure this out with a few more online searches.

1. Relationships
   1. URIs
      1. <https://code.tutsplus.com/articles/sql-for-beginners-part-3-database-relationships--net-8561>
      2. <http://www.tech-recipes.com/rx/56738/one-to-one-one-to-many-table-relationships-in-sql-server/>
      3. <https://docs.microsoft.com/en-us/sql/ssms/visual-db-tools/map-many-to-many-relationships-visual-database-tools>
   2. A one-to-one relationship between two SQL tables is a relationship in which each record in one table has at most one complementary record in another (specific) table that is related to it (the first discussed record) in some way in the real world. One-to-one is the simplest type of relationship to implement. It is important because, in many cases, objects have specific attributes that also fit in a table different from its home table, and thus need to be linked together somehow (with this relationship type). A major concern with this relationship type is that it is not majorly versatile, and so it should be used when there is a discrete relationship between the tables’ entries.

One-to-many relationships are relationships in which each record in one table has zero, one, or many companion records in another (specific) table that it is related to. Many-to-many relationship are the most commonly-used type of relationship, and this is because they are very practical and versatile. Something to keep in mind with this type of connection between records is that records in the ‘many’ side of the relationship can belong to no more than one record on the ‘one’ side.

Many-to-many relationships are the most complex both conceptually and in implementation. They require ‘linking tables’ which are tables that link each record in one of the tables (table A) to many records from the other (table B), and link each record from the second table (table B) to many from the first (table A). Put differently, this means that each record in table A will have as many entries in table AB (the linking table) as it has related records in table B, and vice versa.

* 1. A major ‘gotcha’/notable idea to keep in mind is that many-to-many relationships can become very complex very quickly, and so making a many-to-many-to-many (etc.) may become necessary at times, in order to reduce the complexity of each relationship between two particular tables (and thus to reduce the complexity (but not the length) of a potential query to find some data).

The other most important topic in SQL relationships to keep in mind is that one-to-many (as well as many-to-one, which are pretty much the same concept) are the most commonly-used links between records and for good reason. Their versatility is derived from their definition; each record in table A can have an indeterminate number of attributes from table B, and these attributes for the record from table A are unique to the record itself. This applies to many scenarios and is a good tool for a moderately complex problem in data aggregation.

1. Keywords
   1. URIs
      1. <https://www.codecademy.com/articles/sql-commands>
      2. <https://msdn.microsoft.com/en-us/library/bb208930(v=office.12).aspx>
      3. <http://www.dofactory.com/sql/having>
   2. The SELECT keyword appears in a large number of queries that are run, in general. It is used to retrieve data from the database, most often used with multiple other keywords. It is important because retrieving data is perhaps the most important job regarding databases.  
      The FROM keyword is often used in conjunction with SELECT; it specifies the table containing the desired data to be retrieved. Without it, the database would not know specifically where to find the data and so it has to use some other method of searching for it (that is different from finding the table explicitly).

The WHERE keyword is used to give specific conditions within the query when retrieving data. It filters out rows from the data that was found based on the condition that follows it. It is important because it provides a way to find data that fits specific requirements.  
GROUP BY is used in SELECT statements when aggregating the retrieved data in some way, and then grouping that data according to what follows it. It provides a way to organize the data that was retrieved in a specified manner. This is important especially in large scales because it allows for enhanced clarity/mutual association of the retrieved data points.

HAVING is very similar to WHERE in that it filters out data, but instead of filtering as the data is being retrieved, it filters after the data has been grouped (with GROUP BY). This is useful to us because it allows the data to be filtered according to another condition, often regarding some aggregate function, and so it can be important especially in the cases where data records are being compared against each other.

* 1. A major idea with these keywords is that some of them cannot appear without the others. This reinforces the idea that these keywords and their eventual clauses must form a proper statement in order to be used soundly. WHERE, for example, is not typically used without FROM, and FROM is not typically used without SELECT. The same holds for HAVING typically requiring a usage of GROUP BY.

Even more important is, once again, the extremely common usage of SELECT. Because it is the query type that is most commonly used, it is important to become very familiar with it as early on as possible. Wielding it properly, and with the right companion keywords, can help us to produce effective and efficient queries that get it right, ideally on the first try.