**Querying Microsoft SQL Server 2012 (Exam 70-461) Notes** – by Justin Doh (Last Update 10/2/2016)

Ch1. Foundation of Querying

Lesson1: Foundation of T-SQL

* Mathematical branches that the relational model is based on – Theory and Predicate logic
* Predicate – is an expression that when attributed to some object, makes a proposition either true or false. The relational model uses predicates as one of its core elements.
* Difference between T-SQL and SQL: SQL is standard; T-SQL is the dialect of and extension to SQL that MS implements in its RDBMS – SQL Server.
* Two aspects in which T-SQL deviates from the relational model: 1) A relation has a body with a distinct set of tuples. A table doesn’t have to have a key. T-SQL allows referring to ordinal positions of columns in the ORDER by clause.
* Tuples - (what SQL attempts to represent with rows), A tuple uniquely identifies a slice of data from a cube. The tuple is formed by a combination of dimension members, as long as there are no two or more members that belong to the same hierarchy.
* “Field” and “record” describes physical things, whereas “columns” and “rows” are logical elements of a table.
* “Null value” is incorrect because NULL isn’t a value; it is a mark for a missing value.

Lesson2: Understanding Logical Query Processing

* Difference between WHERFE and HAVING clauses: The WHERE clause is evaluated **before** rows are grouped, and therefore is evaluated per row. The HAVING clause is evaluated **after** rows are grouped, and therefore is evaluated per group.
* You are not allowed to refer to a column alias defined by the SELECT clause in the WHERE clause because the WHERE clause is logically evaluated in a phase **earlier** to the one that evaluates the SELECT clause.
* You are not allowed to refer to a column alias defined by the SELECT clause in the same SELECT clause because all expression that appear in the same logical query processing phase are evaluated conceptually at the same point in time.
* Logical query processing is the conceptual interpretation of the query that defines the correct result, and unlike the keyed-in order of the query clauses, it starts by evaluating the FROM clause.
* Logical query process – It starts with the FROM clause and then moves on WHERE, GROUP BY, HAVING, SELECT and ORDER BY

Ch2. Getting Started with Select statement

Lesson1: Using the FROM and SELECT clauses

* What are the forms of aliasing an attribute in T-SQL? The forms are <expression> AS <alias>, <expression> <alias>, and <alias> = <expression>.
* The FROM clause is the first clause to be logically processed in a SELECT query. In this clause, you indicate the tables you want to query and table operators. You can alias tables in the FROM clause with your chosen names and then use the table alias as a prefix to attribute names.
* Irregular identifier: An identifier that does not follow the rules for formatting identifiers; for example, it starts with a digit, has an embedded space, or is a reserved T-SQL keyword. If you use irregular identifier, delimiters are required.
* Delimiters: For example: “2006” or [2006].

Lesson2: Working with Data Types and Built-In Functions

* Use of the type FLOAT: is an approximate data type and cannot represent all values precisely.
* The NEWID function generates GUID values in random order, whereas the NEWSEQUENTIALID function generates GUIDs that increase in a sequential order.
* SYSDATETIME function returns the current date and time value as a DATETIME2 type.
* Difference between the plus (+) and the CONCAT function: (+) operator by default yields a NULL result on NULL input, whereas the CONCAG function treats NULLs as empty strings.

Ch3. Filtering and Sorting Data

Lesson1: Filtering Data with Predicates

* Predicates: ON, WHERE and HAVING clauses. Predicate is a logical expression. When NULLs are not possible in the data, the predicate can evaluate to true or false.
* COALESCE – Evaluates the arguments in order and returns the current value of the first expression that initially does not evaluate to NULL.

Comparing COALESCE and ISNULL

The ISNULL function and the COALESCE expression have a similar purpose but can behave differently.

1. Because ISNULL is a function, it is evaluated only once. As described above, the input values for the COALESCE expression can be evaluated multiple times.
2. Data type determination of the resulting expression is different. ISNULL uses the data type of the first parameter, COALESCE follows the CASE expression rules and returns the data type of value with the highest precedence.
3. The NULLability of the result expression is different for ISNULL and COALESCE. The ISNULL return value is always considered NOT NULLable (assuming the return value is a non-nullable one) whereas COALESCE with non-null parameters is considered to be NULL. So the expressions ISNULL(NULL, 1) and COALESCE(NULL, 1) although equivalent have different nullability values. This makes a difference if you are using these expressions in computed columns, creating key constraints or making the return value of a scalar UDF deterministic so that it can be indexed as shown in the following example.

* Performance benefit in using the WHERE filter is the it reduces network traffic by filtering in the database server instead of in the client, and one can potentially use indexes to avoid full scans of the tables involved.
* A *search argument*, or SARG, is the form of filter predicate that can rely on index ordering.

Lesson2: Sorting Data

* You guarantees the order of the rows in the result of a query by adding an ORDER BY clause.
* Difference between the result of a query with and one without an ORDER BY clause is without an ORDER BY clause, the result is relational (from an ordering perspective); with an ORDER BY clause, the result is conceptually what the standard calls a Cursor.
* Even when an ORDER BY clause is specified, the result could still have nondeterministic ordering. For deterministic ordering, the ORDER BY list must be unique.

Lesson3: Filtering Data with TOP and OFFSET-FETCH

* By either returning all ties by using the WITH TIES option or by defining unique ordering to break ties, you could guarantee deterministic results with TOP.
* Benefits of using OFFSET-FETCH over TOP: OFFSET-FETCH is standard and TOP isn’t. OFFSET-FETCH supports a skipping capability that TOP doesn’t.
* The ORDER BY clause that is normally used in the query for presentation ordering is also used by TOP and OFFSET FETCH to indicate which rows to filter.
* With the TOP and OFFSET-FETCH options, you could filter data based on a specified number of rows and ordering.

Ch4. Combining Sets

Lesson1: Using Joins

* Difference between old and new syntax for cross joins – the new syntax has the CROSS JOIN keywords between the table names and the old syntax has a comma.
* Three different types of outer joins – left, right and full

Lesson2: Using Subqueries, Table Expressions, and the APPLY operator

* Difference between **self-contained** and **correlated subqueries** – self-contained subqueries are independent of outer query, whereas correlated subqueries have a reference to an element from the table in the outer query.
* Difference between the APPLY and JOIN operators – With JOIN operator, both inputs represent static relations. With APPLY, the left side is a static relation, but the right side can be a table expression with correlations to elements from the left table.
* T-SQL supports four kinds of table expressions, which are named query expressions.
  1. Derived tables and (2) CTEs are types of table expressions that are visible only in the scope of the statement that defined them. (3) Views and

(4) Inline table-valued functions are reusable table expressions whose definitions are stored as objects in the database. Views do not support input parameters, whereas inline table-valued functions do.

- Apply operators – two operator – CROSS and OUTER; CROSS APPLY operator operates on left and right table expressions as inputs.

OUTER APPLY operator includes in the result rows from the left side that get an empty set back form the right side.

Lesson3: Using Set Operators

* Which set operators does T-SQL support? Union, Intersect and Except set operators, as well as the UNION ALL multiset operator. The Except operator performs set difference.
* The UNION ALL operator unified the inputs without eliminating duplicates. The UNION operator unifies the input sets, returning distinct rows.
* Two requirements for the queries involved in a set operator – The number of columns in the two queries needs to be the same, and corresponding columns need to have compatible types.

Ch5. Grouping and Windowing

Lession1: Writing Grouped Queries

* An aggregate function, a GROUP BY clause, makes a query a grouped query
* GROUPING SETS, CUBE and ROLLUP make multiple grouping sets in the same query.
* You can apply aggregate functions to the groups, such as COUNT, SUM, AVG, MIN and MAX to the groups.
* You could use newer features in the language to define multiple grouping sets in one query by using the GROUPING SETS, CUBE, and ROLLUP clauses.

Lesson2: Pivoting and Unpivoting Data

* Difference between PIVOT and UNPIVOT – PIVOT rotates data from a state of rows to a state of columns; UNPIVOT rotates the data from columns to rows.
* When you pivot data, you need to identify three things: the grouping element, spreading element, and aggregation element.
* Pivoting is a special form of grouping and aggregating data where you rotate data from a state of rows to a state of columns.
* To unpivot data, you need to identify three things: the source columns that you need to unpivot, the target names column, and the target values column.
* Type of language constructs the PIVOT and UNPIVOT implemented as: table operators.

Lesson3: Using Window Functions

* Partitioning, ordering, and framing clauses are the three different types of window function.
* The delimiters UNBOUNDED PRECEDING and UNBOUNDED FOLLOWING represent the beginning and end of the partition.
* Window functions perform data analysis computations. They operate on a set of rows defined for each underlying row by using a clause called OVER.
* T-SQL supports window aggregate, ranking and offset functions. All window functions support window partition and window order clauses. Aggregate window functions, in addition to FIRST-VALUE and LAST-VALUE, also supports a window frame clause.

Ch6. Querying Full-Text Data

Ch7. Querying and Managing XML Data

Ch8. Creating Tables and Enforcing Data Integrity

Ch9. Designing and Creating Views, Inline Functions, and Synonyms

Ch10. Inserting, Updating, and Deleting Data