CIS560

Single-Table Queries - Part 1

Topics

- Schemas & Objects
- Minimum Table Requirements
- DDL vs. DML
- Booleans
- SELECT Statement
  - SELECT
  - FROM
  - WHERE
  - GROUP BY
  - HAVING

SQL Server Schemas & Objects

- Almost everything inside a database is an object such as a table, view, procedure, function, and index.
- A schema is a container that holds objects, similar to a namespace.
- A schema cannot hold other schemas.
- A fully-qualified object name includes the schema name:

```sql
SELECT *
FROM Sales.Orders;
```

Table Requirements

- Table
  - Each table belongs to a schema
  - Has one or more columns
- Minimum column requirements
  - Unique name within the table
  - Data type
  - Nullability (NULL vs. NOT NULL)

Query Basics

- SQL is a declarative language
  - Define what you want, not how to get it.
- SQL is a set-oriented language
  - Not procedural like C# or other programming languages.
  - Order rarely matters to accomplish what you want.
- What is the most common problem with beginners' solutions?
  - They disregard one or both of these properties.

Query Basics: DDL vs. DML

- Data Manipulation Language (DML)
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - MERGE
- Data Definition Language (DDL)
  - CREATE | ALTER TABLE
  - CREATE | ALTER VIEW
  - …

SELECT Element

- SELECT is the only element required in a SELECT statement.
- Usually, though, you have SELECT and FROM
- Column aliasing is supported, at times required
- The SELECT element provides support for the **projection** operation.

See the docs here

FROM Element

- FROM defines which table(s)
- In other words, where to get the data
- Table aliasing is supported
- Resulting column names in SELECT element inherit the column names from table

WHERE Element

- WHERE provides filtering.
- Accepts any predicate or Boolean expression
- The WHERE element provides support for the **selection** operation.

See the docs here

Booleans

- Booleans are only supported as expressions
  - There is no Boolean data type.
  - They cannot be stored in a variable, column, or result set.
- What are they used for?
  - WHERE clauses
  - Control statements such as IF and WHILE
  - CASE function (SQL’s switch statement)
- Comparison operators:
  - Standard: =, >, <, >=, <=, <>
  - !=, !<, !> supported but not ISO standard

Booleans (cont.)

- QUIZ: How many resulting values are possible for a boolean expression?

  Answer

  Three: TRUE, FALSE, and UNKNOWN

- See Example

GROUP BY Element

- Defines a distinct set by one or more columns or expressions
- Allows aggregated computation over that set.
- ANSI Elements
  - Simple column list
  - Expressions
  - GROUPING SETS
  - CUBE
  - ROLLUP
- Can be combined with aggregates such as MAX, MIN, AVG, COUNT, etc.

GROUP BY Element - Aggregates

- Almost all aggregate functions have this syntax:

```
<Function Name> ( [ ALL | DISTINCT ] expression )
```

- Examples are MIN, MAX, AVG, and SUM.
- COUNT and COUNT_BIG are similar but allow for no expression (*).

```
COUNT ( { [ [ ALL | DISTINCT ] expression ] | * } )
```

The asterisk (*) means just count the rows.

- Can be used with the OVER clause to define partitions.

```
OVER ( [ PARTITION BY value_expression , ... [ n ] ] )
```

- Null values are ignored.

HAVING Element

- Provides a post-grouping filter
- Like WHERE, accepts any Boolean expression
- Aggregated computations can be used in the filter

SELECT Statement Processing Order

- Major elements of SELECT

- ANSI Processing Order (Logical)

```
5   SELECT ...
1   FROM ...
2   WHERE ...
3   GROUP BY ...
4   HAVING ...
```

Syntax

```
SELECT select_list
[ FROM table ]
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
```

Examples

```sql
SELECT O.OrderID, O.OrderDate,
   O.CustomerID, O.SalespersonPersonID
FROM Sales.Orders O
WHERE O.OrderDate >= '2016-01-01'
   AND O.OrderDate < '2017-01-01'
```

```sql
SELECT YEAR(O.OrderDate) AS OrderYear,
   COUNT(*) AS OrderCount
FROM Sales.Orders O
GROUP BY YEAR(O.OrderDate)
HAVING COUNT(*) > 20000;
```

Questions?