Fully Qualified Domain

- User defined data type which are:
    - Can specify data type (INT, VARCHAR, etc)
    - Defined once
    - Defined for the entire database
    - Reusable for entire database tables
        - Like CustomerID in Sales.Customer and Sales.Order (standardized)
    - Creates uniformity with the data
- Standardization leads to consistency → make more sense out of data
- Has constraints such as having a specific character in a string, like a ‘,’ in a FullName column to separate first and last name
- SQL Server: CREATE TYPE,
  ANSI SQL: CREATE DOMAIN
    - Both create rules and constraints; for the purpose of consistency
    - CREATE TYPE used with schema namesapce
- Enforces format

Taxonomy

- Definition(gen.): involved with the classification of things (ie the classification of the animal kingdom
- SQL context: creates a structured classification of database
- Grouping logically: Sales, HR
- FQDS will have consistent naming and rules across the database to make it all uniform and understandable
- Standardized domains in a taxonomy are reused across the data (governance)

Fully Qualified Table Name

- FQTN provides a complete path to a table in a database
- Format: database_name.schema_name.table_name
- Uses:
    - Avoids collision and removes ambiguity. If tables or columns share the same name within a database, a FQTN will help to clear it up
    - Multi-layer precision using: database, schema then table which promotes precision → provides accurate table being queried
    - Precision → pinpoint the exact table a user wants, also doesn’t get mixed up with other names
    - Prevent any unexpected results
    - Helps to “document” code
    - FOREIGN key from CHILD table accesses PRIMARY key from PARENT table (referencial integrity)
    - Heterogeneous database: diff systems/platforms
    - FQTN helps by: eliminating conflicts, so you can add additional databases without messing up functionality/be specific about the address