My role in the presentation was to explain the difference between ANSI SQL's `CREATE DOMAIN` vs T-SQL's `CREATE TYPE`. I also introduced the concept of fully qualifiued domains and their use cases. The following are the notes I've taken on the topics asked in the assignment.

Taxonomies

The literal definition of a **taxonomy** is "a system of classification". In relational database design taxonomies are a way to classify and organize data into categories. These categories can be general or specific depending on the use case

Example: Stock Item Categories

- A real world example is to sort products on a store into categories so customers can find what they want easily
- In WorldWideImporters, there is the table `Warehouse.StockGroups` which groups items based on their type. This is the taxonomy.
- The items in `Warehouse.StockItems` can be joined by `Warehouse.StockItemStockGroups` to obtain the specific groups an item falls in
    - For example, Shirts and pants go into clothing while boxes and tape go into packing materials.
- In this case some items can actually belong to multiple categories.

Fully Qualified Domains (FQDs)

A **domain** in SQl is a data type with set of valid values in a table's columns. Some of these constraints can be the data type allowed in the collumn, checks, nullability, and default values.

For example:

- You cant have letters in a price, so it is either an `INT` or `FLOAT`
- If you don't allow anyone under 18, a simple `CHECK (Age >= 18)` means that in no situation someone underaged can be added to your table.

A **Fully Qualified Domain** is a domain that is called by its schema. It connects directly to the exact place in the database. In a situation where domain names can be similar to each other, a FQD can make sure you are using the correct domain. Once a FQD is defined, each time it is used the same constraints are applied. Any changes to a domain will also be implemented to every other instance ensuring consistency and reliability.

Fully Qualified Table Names (FQTNs)

Similarly to a FQD, a **fully qualified table name** is a way to explicitly call a specific table. This prevents confusion between tables with the same name. A clear example is looking for customers, a database may have many tables with the name `Customers`

- `dbo.Customers`
- `Sales.Customers`
- `Messages.Customers`

- `Customers` (the schema)

This allows for integrity as you are not mixing up data from another table, and a FQTN is required for joining tables. FQTNs can also be used to link together tables from different servers and databases, allowing scalability for larger and complex systems.

ANSI vs T-SQL

ANSI SQL `CREATE DOMAIN`:

- ANSI SQL supports creating domains. These domains have pre defined constraints and can be reused across table columns.
- This creates a single object that bundles the data type AND the rule together.

```
CREATE DOMAIN WWI.PhoneNumber
AS NVARCHAR(20)
CHECK (VALUE LIKE '+[0-9]%');
```

T-SQL `CREATE TYPE`:

- T-SQL Does not support creating domains.
- A type is still a reusable object with constraints, but does not support `CHECK` inside a type
- The `CHECK` constraint must be added to *every single table* that uses the type

```
-- Create a Type
CREATE TYPE dbo.PhoneNumber
FROM NVARCHAR(20);

-- Use the type in a table
CREATE TABLE dbo.PhoneNumber(
    CHECK (VALUE LIKE '+[0-9]%')
)
```

Sources

1. https://amplitude.com/explore/data/what-data-taxonomy
2. https://docs.teradata.com/r/Enterprise_IntelliFlex_VMware/Database-Design/The-Activity-Transaction-Modeling-Process/Domains
3. https://www.tektutorialshub.com/sql-server/fully-qualified-table-names-in-sql-server/
4. https://learn.microsoft.com/en-us/sql/t-sql/statements/create-type-transact-sql?view=sql-server-ver17
5. https://www.sololearn.com/en/Discuss/1390307/what-does-fully-qualified-name-means-in-sql