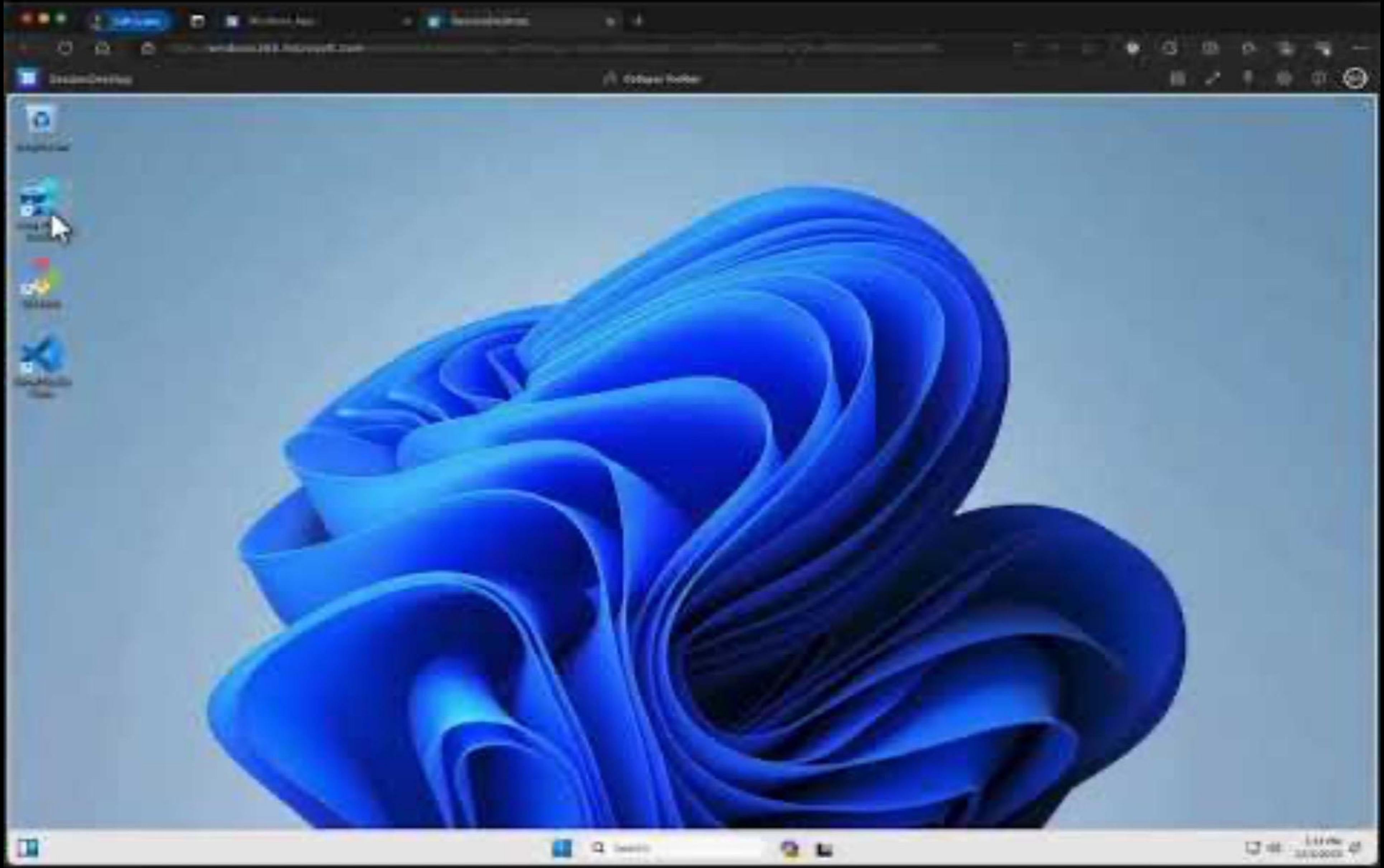
Microsoft

1: Getting started with Transact-SQL

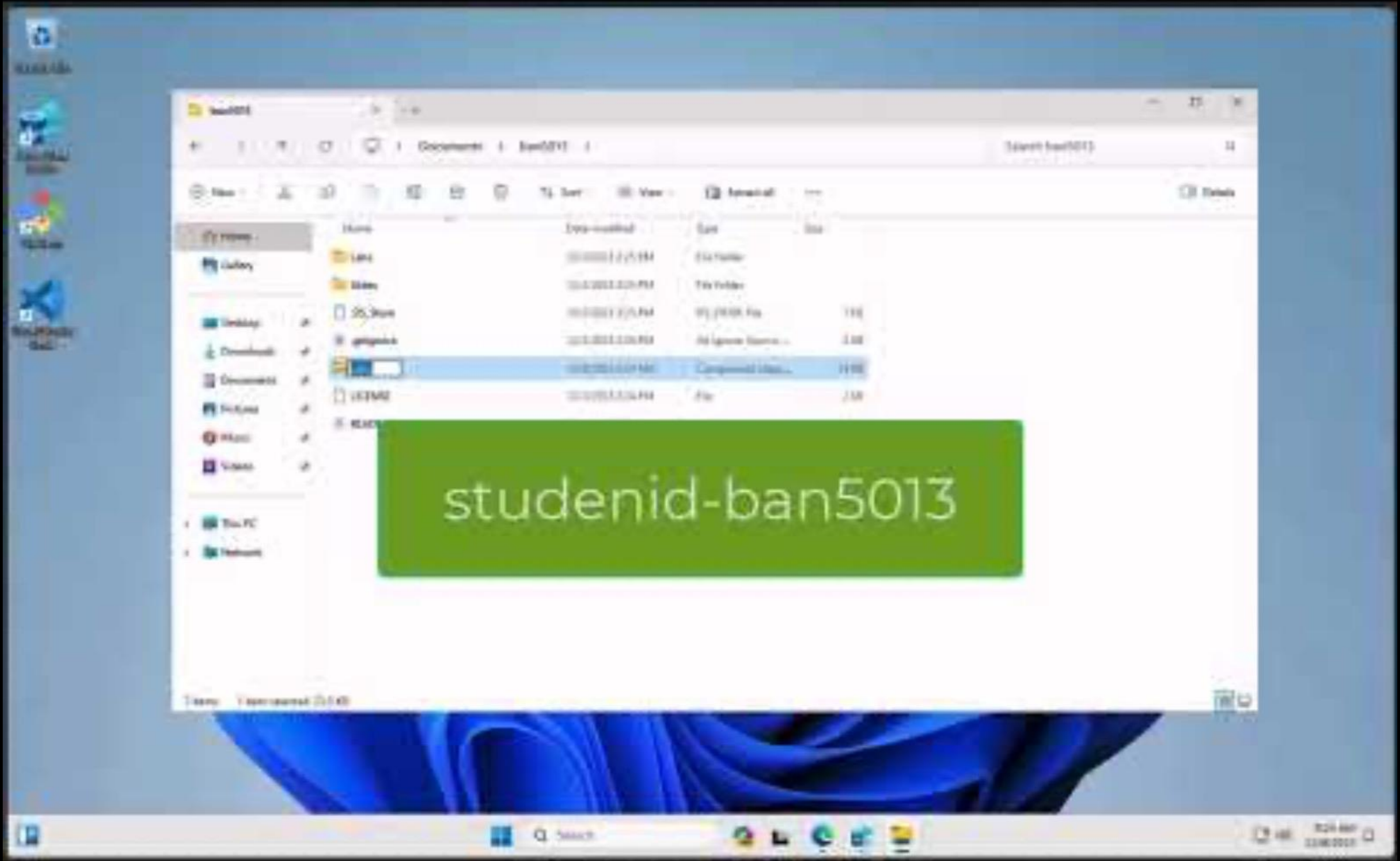
https://youtu.be/ob8zvndqHc0

Lab Setup



https://youtu.be/PF4nnzmiam8

Lab Submission



Agenda

- Introduction to Transact-SQL
- Using the SELECT Statement

1: Introduction to Transact-SQL

What is Transact-SQL?

Structured Query Language (SQL)

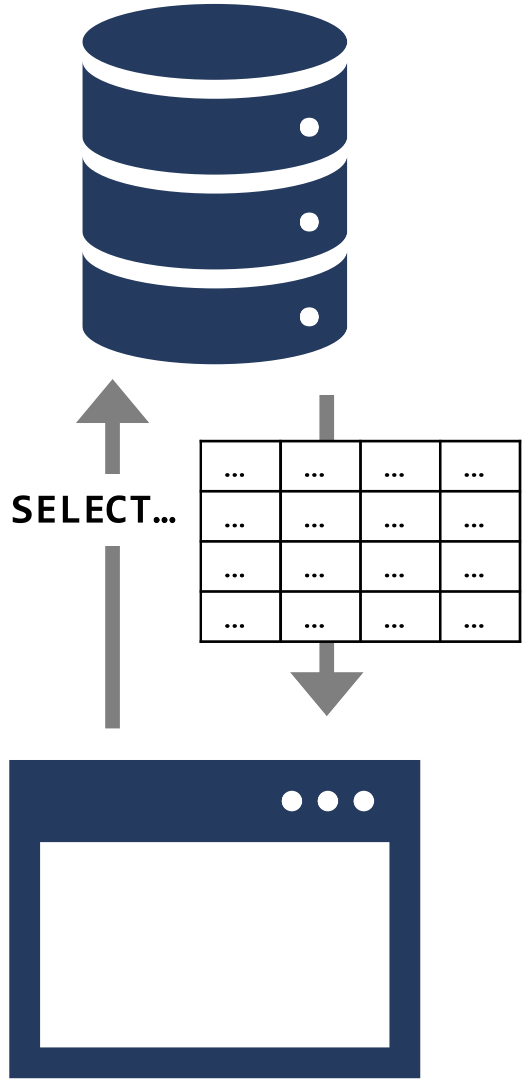
- Developed in the 1970s as a language for querying databases
- Adopted as a standard by ANSI and ISO standards bodies
- Widely used across multiple database systems

Microsoft's implementation is Transact-SQL

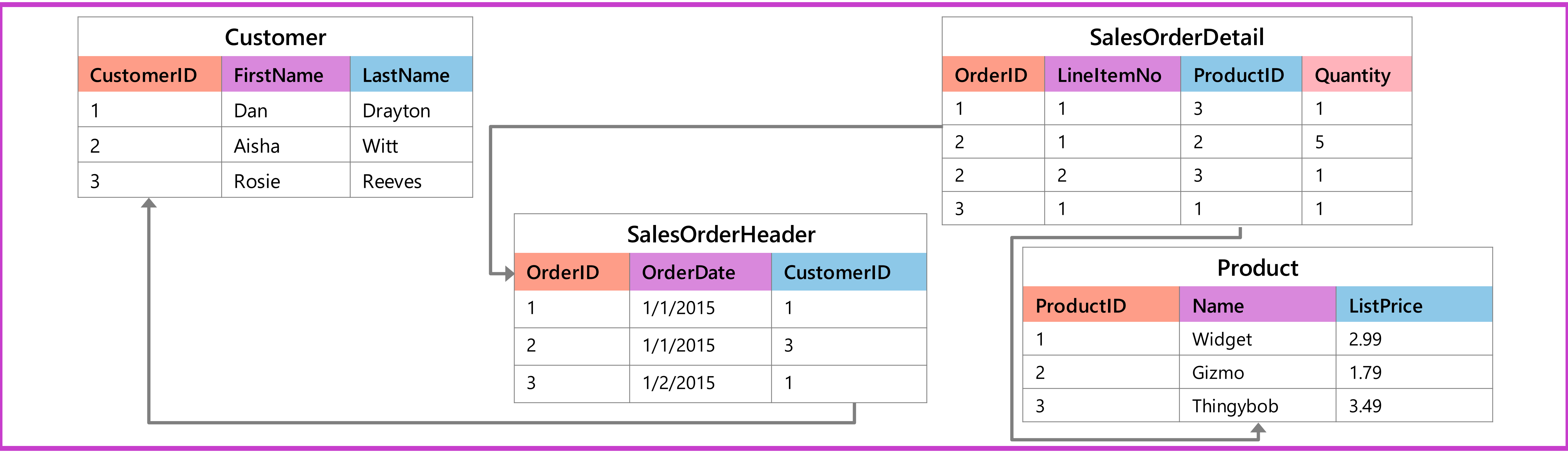
- Often referred to as T-SQL
- Query language for SQL Server, Azure SQL Database, and other Microsoft relational database services

SQL is *declarative*, not *procedural*

- Describe what you want, don't specify steps



Relational databases



Customer

CustomerID	FirstName	LastName
1	Dan	Drayton
2	Aisha	Witt
3	Rosie	Reeves

SalesOrderDetail

OrderID	LineItemNo	ProductID	Quantity
1	1	3	1
2	1	2	5
2	2	3	1
3	1	1	1

SalesOrderHeader

OrderID	OrderDate	CustomerID
1	1/1/2015	1
2	1/1/2015	3
3	1/2/2015	1

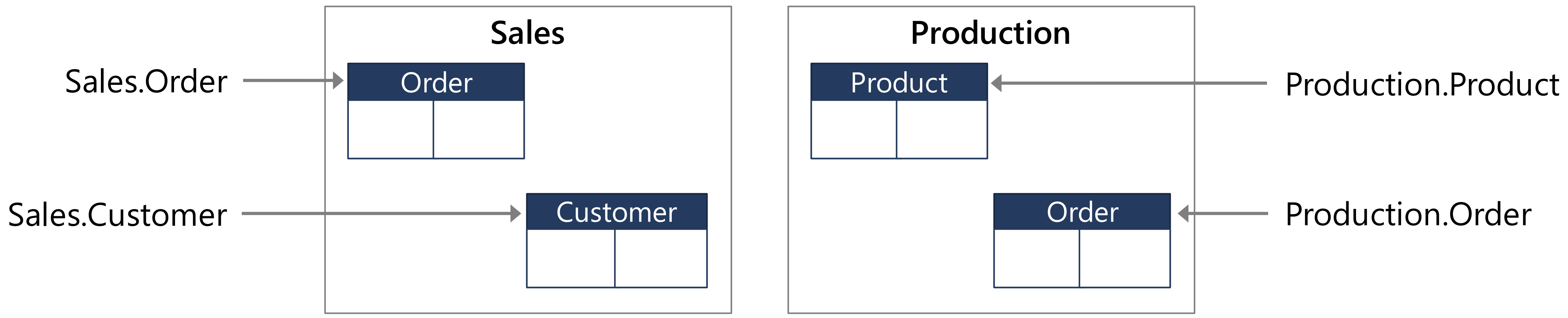
Product

ProductID	Name	ListPrice
1	Widget	2.99
2	Gizmo	1.79
3	Thingybob	3.49

Entities are represented as *relations* (tables), in which their attributes are represented as domains (columns)

Most relational databases are *normalized*, with relationships defined between tables through primary and foreign *keys*

Schemas and object names



Schemas are namespaces for database objects

- Fully-qualified names:

  [*server_name*.][*database_name*.][*schema_name*.]*object_name*

- Within database context, best practice is to include schema name:

  schema_name.object_name

SQL statement types

Data Manipulation Language (DML)	Data Definition Language (DDL)	Data Control Language (DCL)
Statements for querying and modifying data: • SELECT • INSERT • UPDATE • DELETE	Statements for defining database objects: • CREATE • ALTER • DROP	Statements for assigning security permissions: • GRANT • REVOKE • DENY

Focus of this course

2: Using the SELECT statement

The SELECT statement

	Element	Expression	Role
5	SELECT	<select list>	Defines which columns to return
1	FROM	<table source>	Defines table(s) to query
2	WHERE	<search condition>	Filters rows using a predicate
3	GROUP BY	<group by list>	Arranges rows by groups
4	HAVING	<search condition>	Filters groups using a predicate
6	ORDER BY	<order by list>	Sorts the output

```
SELECT OrderDate, COUNT(OrderID) AS Orders

FROM Sales.SalesOrder

WHERE Status = 'Shipped'

GROUP BY OrderDate

HAVING COUNT(OrderID) > 1

ORDER BY OrderDate DESC;
```

Basic SELECT query examples

All columns

```
SELECT * FROM Production.Product;
```

Specific columns

```
SELECT Name, ListPrice
FROM Production.Product;
```

Expressions and aliases

```
SELECT Name AS Product, ListPrice * 0.9 AS SalePrice
FROM Production.Product;
```

Data types

Exact numeric	Approximate numeric	Character	Date/time	Binary	Other
tinyint	float	char	date	binary	cursor
smallint	real	varchar	time	varbinary	hierarchyid
int		text	datetime	image	sql_variant
bigint		nchar	datetime2		table
bit		nvarchar	smalldatetime		timestamp
decimal/numeric		ntext	datetimeoffset		uniqueidentifier
numeric					xml
money					geography
smallmoney					geometry

- Compatible data types can be implicitly converted
- Explicit conversion requires an explicit conversion function:

```
CAST / TRY_CAST
CONVERT / TRY_CONVERT
PARSE / TRY_PARSE
STR
```

NULL values

NULL represents a *missing* or *unknown* value

ANSI behaviour for NULL values:

- The result of any expression containing a NULL value is NULL

```
2 + NULL = NULL
'MyString: ' + NULL = NULL
```

- Equality comparisons (=) always return false for NULL values, use IS NULL

```
NULL = NULL returns false
NULL IS NULL returns true
```

Useful functions:

`ISNULL(`*`column/variable, value`*`)`: Returns *value* if the column or variable is NULL

`NULLIF(`*`column/variable, value`*`)`: Returns NULL if the column or variable is *value*

`COALESCE(`*`column/variable1, column/variable2, ...`*`)`: Returns the value of the first non-NULL column or variable in the list

Lab: Get started with Transact-SQL

- Explore the *AdventureWorks* database
- Use SELECT queries to retrieve data
- Handle NULL values
- Work with data types

Review

1 You must return the *Name* and *Price* columns from a table named *Product* in the *Production* schema. In the resulting rowset, you want the *Name* column to be named *ProductName*. Which of the following Transact-SQL statements should you use?

- ❑ `SELECT * FROM Product AS Production.Product;`
- [✓ `SELECT Name AS ProductName, Price FROM Production.Product;`]
- ❑ `SELECT ProductName, Price FROM Production.Product;`

2 You must retrieve data from a column that is defined as char(1). If the value in the column is a digit between 0 and 9, the query should return it as an integer value. Otherwise, the query should return NULL. Which function should you use?

- ❑ `CAST`
- ❑ `NULLIF`
- [✓ `TRY_CONVERT`]

3 You must return the *Cellphone* column from the *Sales.Customer* table. *Cellphone* is a varchar column that permits NULL values. For rows where the *Cellphone* value is NULL, your query should return the text 'None'. What query should you use?

- [✓ `SELECT ISNULL(Cellphone, 'None') AS Cellphone FROM Sales.Customer;`]
- ❑ `SELECT NULLIF(Cellphone, 'None') AS Cellphone FROM Sales.Customer;`
- ❑ `SELECT CONVERT(varchar, Cellphone) AS None FROM Sales.Customer;`

Microsoft