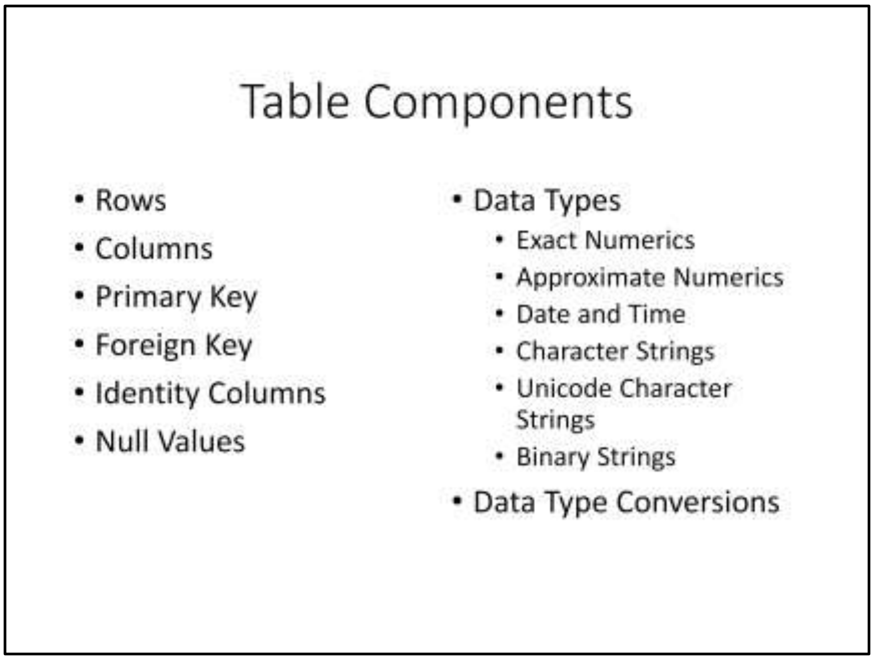
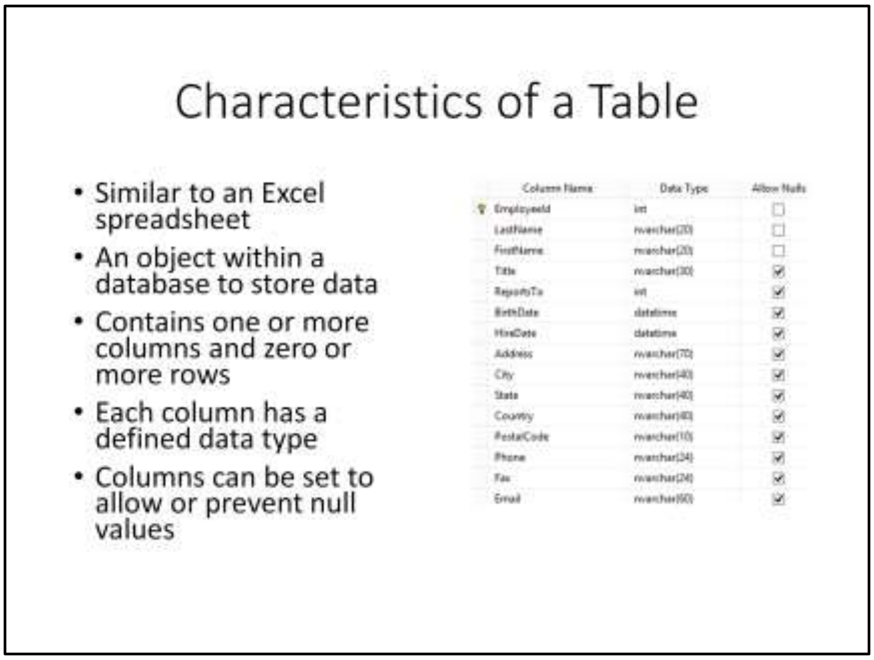


Table Components

- Rows
- Columns
- Primary Key
- Foreign Key
- Identity Columns
- Null Values
- Data Types
  - Exact Numerics
  - Approximate Numerics
  - Date and Time
  - Character Strings
  - Unicode Character Strings
  - Binary Strings
- Data Type Conversions

This presentation will cover the basic components of a SQL Server table. I will talk about rows, columns, primary and foreign keys, identity columns, null values, and the various data types that can be part of a table.

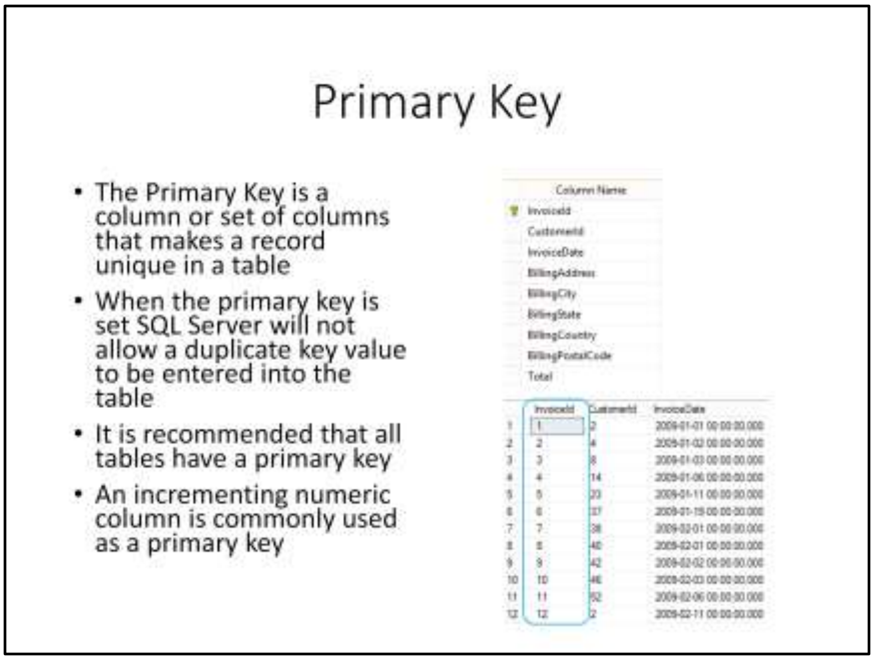


Characteristics of a Table

- Similar to an Excel spreadsheet
- An object within a database to store data
- Contains one or more columns and zero or more rows
- Each column has a defined data type
- Columns can be set to allow or prevent null values

Column Name	Data Type	Allow Nulls
EmployeeId	int	☐
LastName	nvarchar(20)	☐
FirstName	nvarchar(20)	☐
Title	nvarchar(30)	✓
ReportsTo	int	✓
BirthDate	datetime	✓
HireDate	datetime	✓
Address	nvarchar(70)	✓
City	nvarchar(40)	✓
State	nvarchar(40)	✓
Country	nvarchar(40)	✓
PostalCode	nvarchar(10)	✓
Phone	nvarchar(24)	✓
Fax	nvarchar(24)	✓
Email	nvarchar(60)	✓

A SQL Server table looks very similar to an Excel spreadsheet. In fact it is common for database users to export the contents of a SQL Server table into Excel, and vice-versa. A table is a series of one or more columns, and zero or more rows. When a table is created each column needs to be mapped to a specific data type. You must also decide whether each column will allow null values, which means is data required for that column or not. Once the table is defined, you can begin populating the table with rows.



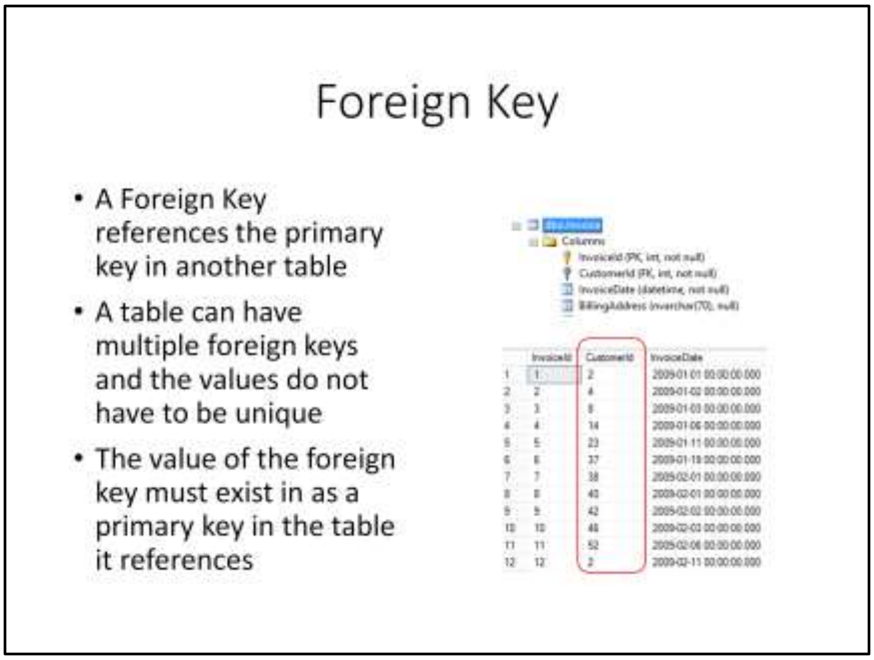
Primary Key

- The Primary Key is a column or set of columns that makes a record unique in a table
- When the primary key is set SQL Server will not allow a duplicate key value to be entered into the table
- It is recommended that all tables have a primary key
- An incrementing numeric column is commonly used as a primary key

Column Name
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

	InvoiceId	CustomerId	InvoiceDate
1	1	2	2009-01-01 00:00:00.000
2	2	4	2009-01-02 00:00:00.000
3	3	8	2009-01-03 00:00:00.000
4	4	14	2009-01-06 00:00:00.000
5	5	23	2009-01-11 00:00:00.000
6	6	37	2009-01-19 00:00:00.000
7	7	38	2009-02-01 00:00:00.000
8	8	40	2009-02-01 00:00:00.000
9	9	42	2009-02-02 00:00:00.000
10	10	46	2009-02-03 00:00:00.000
11	11	52	2009-02-06 00:00:00.000
12	12	2	2009-02-11 00:00:00.000

It is considered good practice to include a primary key in every table you create. A primary key is a column or group of columns that makes each row in the table unique. You will never find duplicate values in a primary key. SQL Server enforces the uniqueness of a primary key by preventing duplicate values from being entered. It is also not allowed for any of the columns in a primary key to be null. A primary key is what makes a record unique from all other records in a table.



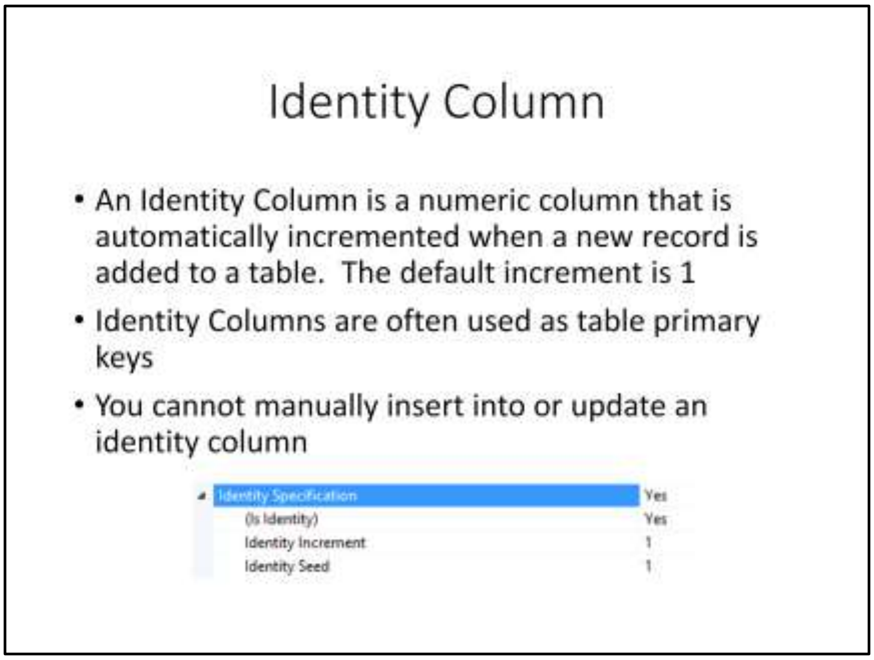
Foreign Key

- A Foreign Key references the primary key in another table
- A table can have multiple foreign keys and the values do not have to be unique
- The value of the foreign key must exist in as a primary key in the table it references

```
dbo.Invoice
  Columns
    InvoiceId (PK, int, not null)
    CustomerId (FK, int, not null)
    InvoiceDate (datetime, not null)
    BillingAddress (nvarchar(70), null)
```

	InvoiceId	CustomerId	InvoiceDate
1	1	2	2009-01-01 00:00:00.000
2	2	4	2009-01-02 00:00:00.000
3	3	8	2009-01-03 00:00:00.000
4	4	14	2009-01-06 00:00:00.000
5	5	23	2009-01-11 00:00:00.000
6	6	37	2009-01-19 00:00:00.000
7	7	38	2009-02-01 00:00:00.000
8	8	40	2009-02-01 00:00:00.000
9	9	42	2009-02-02 00:00:00.000
10	10	46	2009-02-03 00:00:00.000
11	11	52	2009-02-06 00:00:00.000
12	12	2	2009-02-11 00:00:00.000

A foreign key is a column in table that references the primary key of another table. Only values that exist in the primary key table column can be entered into the foreign key column. However it is allowable in some cases for a foreign key to be null. It is also possible for there to be duplicate values in a foreign key column. Primary and foreign keys are the foundation of relational databases. I will talk more about them during my presentation on querying from multiple tables.

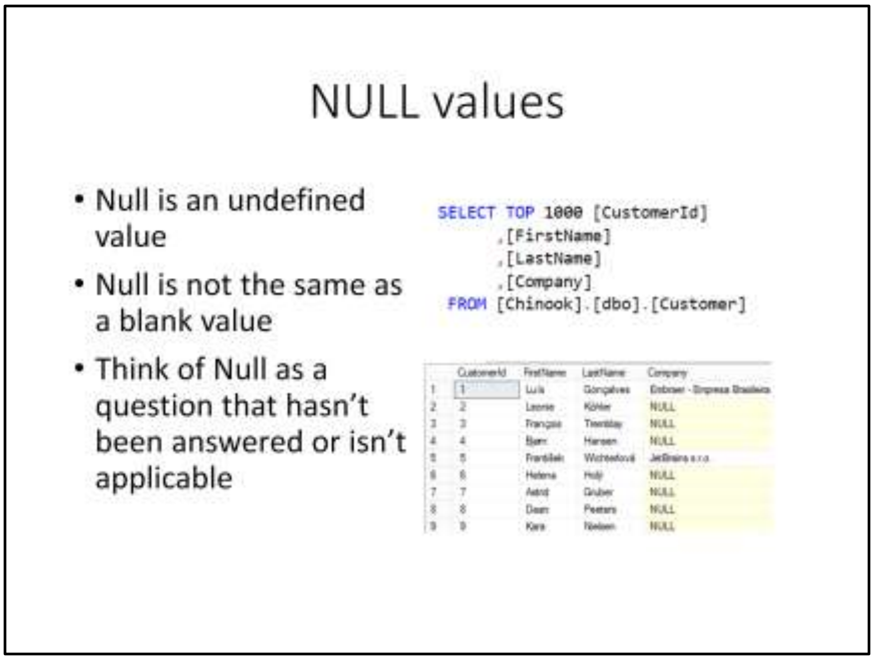


Identity Column

- An Identity Column is a numeric column that is automatically incremented when a new record is added to a table. The default increment is 1
- Identity Columns are often used as table primary keys
- You cannot manually insert into or update an identity column

Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

An identity column is a numeric column that automatically inserts a new integer into itself when a row is added to the table. The default is to always increase the number entered by one. Identity columns often serve as primary keys. They are useful when the records being entered into a table cannot be guaranteed to be unique. It is not possible to manually update an identity column.



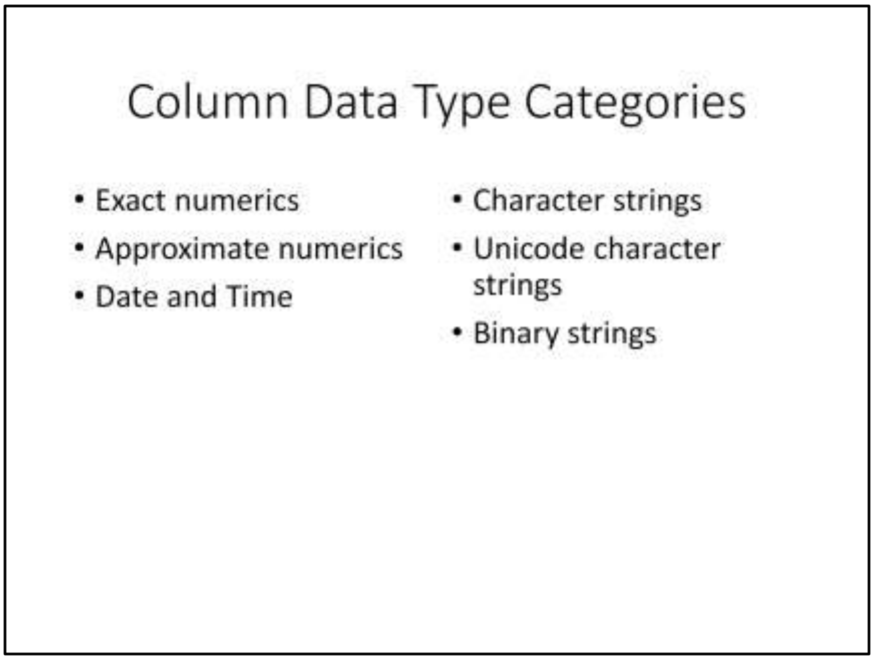
NULL values

- Null is an undefined value
- Null is not the same as a blank value
- Think of Null as a question that hasn't been answered or isn't applicable

```
SELECT TOP 1000 [CustomerId]
      ,[FirstName]
      ,[LastName]
      ,[Company]
  FROM [Chinook].[dbo].[Customer]
```

	CustomerId	FirstName	LastName	Company
1	1	Luís	Gonçalves	Embraer - Empresa Brasileira
2	2	Leonie	Köhler	NULL
3	3	François	Tremblay	NULL
4	4	Bjørn	Hansen	NULL
5	5	František	Wichterlová	JetBrains s.r.o.
6	6	Helena	Holý	NULL
7	7	Astrid	Gruber	NULL
8	8	Daan	Peeters	NULL
9	9	Kara	Nielsen	NULL

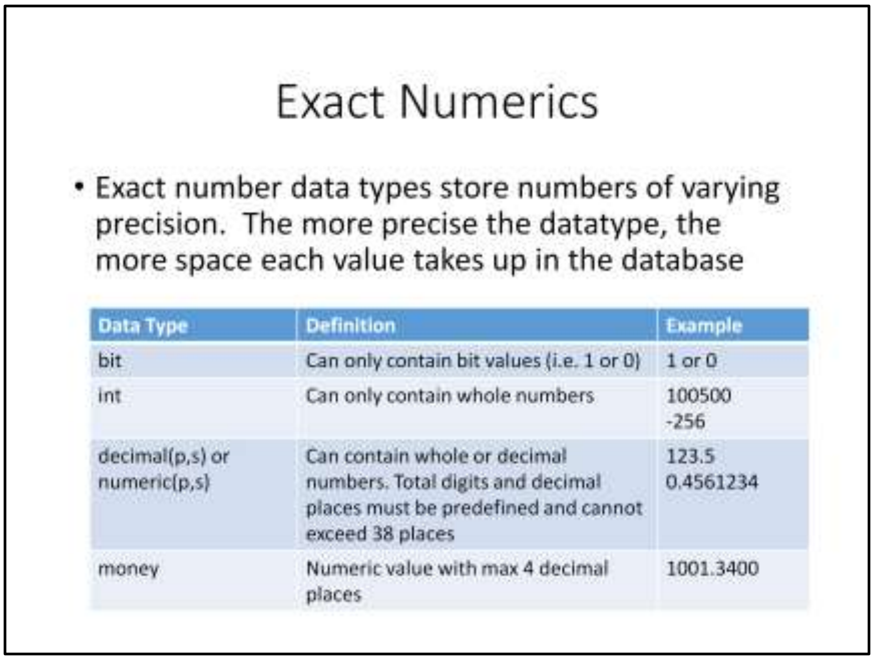
Null is a special condition in SQL Server. It represents an undefined field. It is different from a blank value. For example you could ask someone what is their middle name. If they say they don't have a middle name then a blank value could represent the fact they were asked. If a person wasn't asked, or they didn't answer then you can use null to represent the unknown answer.



Column Data Type Categories

- Exact numerics
- Approximate numerics
- Date and Time
- Character strings
- Unicode character strings
- Binary strings

When a table is created the columns of the table need to have data types assigned to them. The basic categories are numeric data, date-time data, string data, and binary data. Data types determine how data is stored and displayed, as well as how it can be sorted and searched against. There are other possible data types but we won't be covering them in this course.

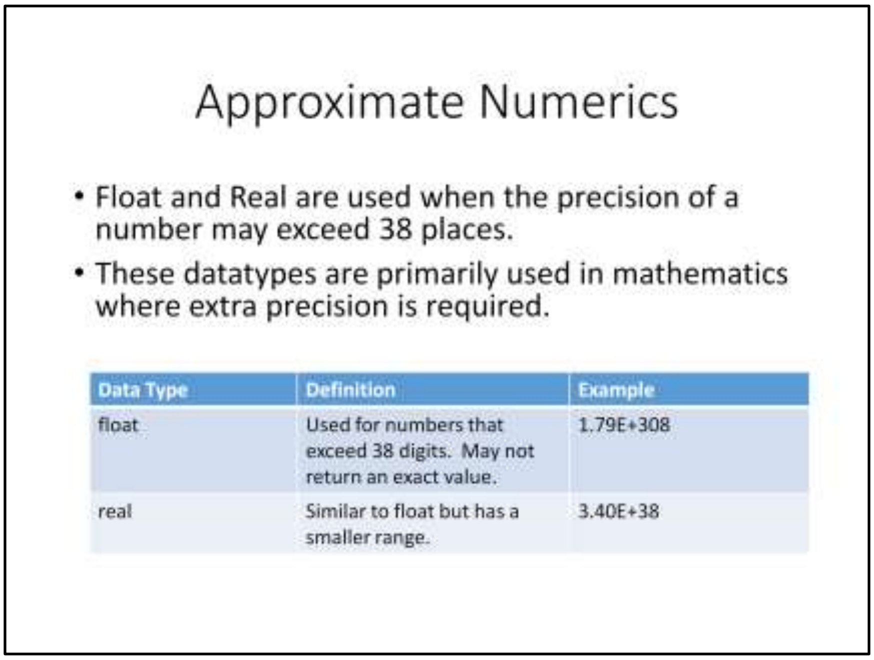


Exact Numerics

- Exact number data types store numbers of varying precision. The more precise the datatype, the more space each value takes up in the database

Data Type	Definition	Example
bit	Can only contain bit values (i.e. 1 or 0)	1 or 0
int	Can only contain whole numbers	100500 -256
decimal(p,s) or numeric(p,s)	Can contain whole or decimal numbers. Total digits and decimal places must be predefined and cannot exceed 38 places	123.5 0.4561234
money	Numeric value with max 4 decimal places	1001.3400

Exact numerics are data types that hold the exact value of a number. The various numeric data types can be used to restrict the type of number that can be stored. For example the bit data type can only hold the number one or zero, and is often used with true/false questions. The integer or int data type only stores whole numbers. Fractions are not allowed. The decimal data type allows the column to store decimal numbers. When creating a decimal data type you need to include the precision and scale in parenthesis. The scale defines the number of digits that can be to the right of the decimal point, and the precision defines the total number of digits both to the left and right of the decimal. The decimal and numeric data types are identical, their names can be interchanged. Money is a special numeric data type that that allows up to four decimal places after the decimal. Money does not store the currency type.

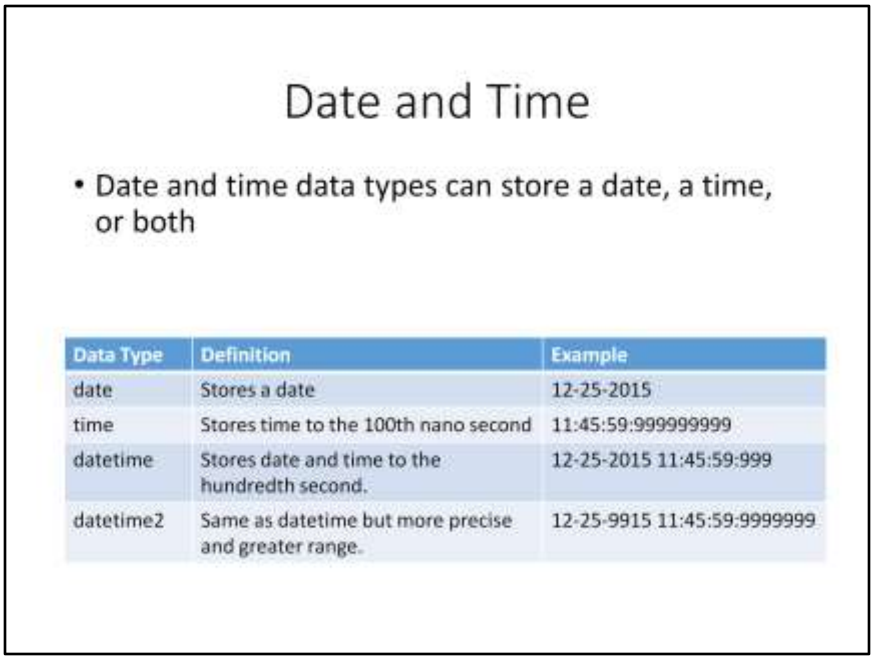


Approximate Numerics

- Float and Real are used when the precision of a number may exceed 38 places.
- These datatypes are primarily used in mathematics where extra precision is required.

Data Type	Definition	Example
float	Used for numbers that exceed 38 digits. May not return an exact value.	1.79E+308
real	Similar to float but has a smaller range.	3.40E+38

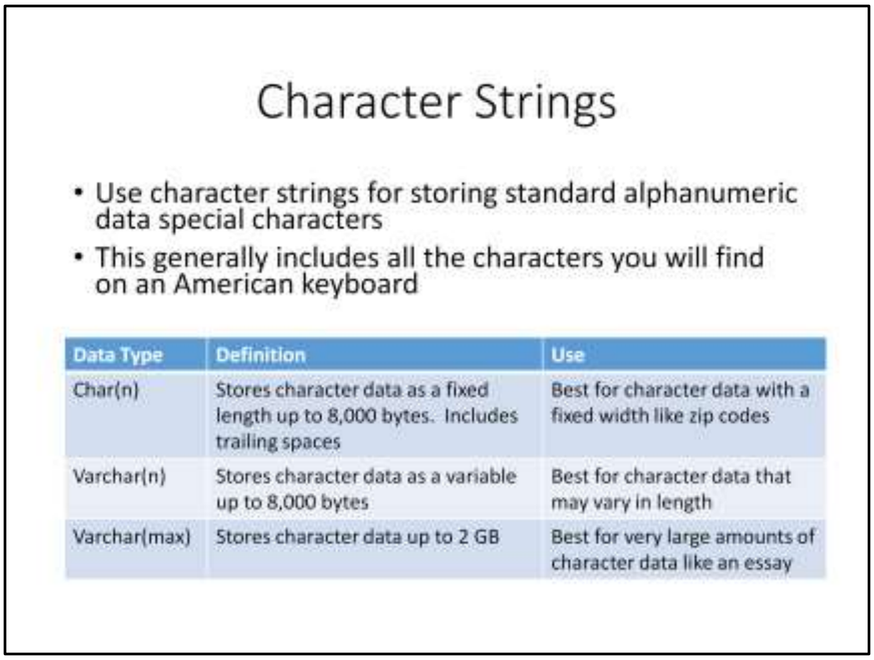
Approximate numerics are used when the numbers being stored can exceed 38 digits. For example if you wanted to compare the weight of a helium atom to the planet Jupiter, you would likely use a float data type to store the result. The float and real data types are often used in mathematics.


Date and Time

- Date and time data types can store a date, a time, or both

Data Type	Definition	Example
date	Stores a date	12-25-2015
time	Stores time to the 100th nano second	11:45:59:999999999
datetime	Stores date and time to the hundredth second.	12-25-2015 11:45:59:999
datetime2	Same as datetime but more precise and greater range.	12-25-9915 11:45:59:9999999

Date and time data type are used to store date and time values. The date data type only stores dates and the time data type only stores time values. The datetime and datetime2 data types both store the date and time together as a single value. Datetime2 is the newer of the two datatypes, and Microsoft recommends using datetime2 over datetime. This is because datetime2 is more precise, has a greater date range, and is compatible with the ranges of other sql server products.

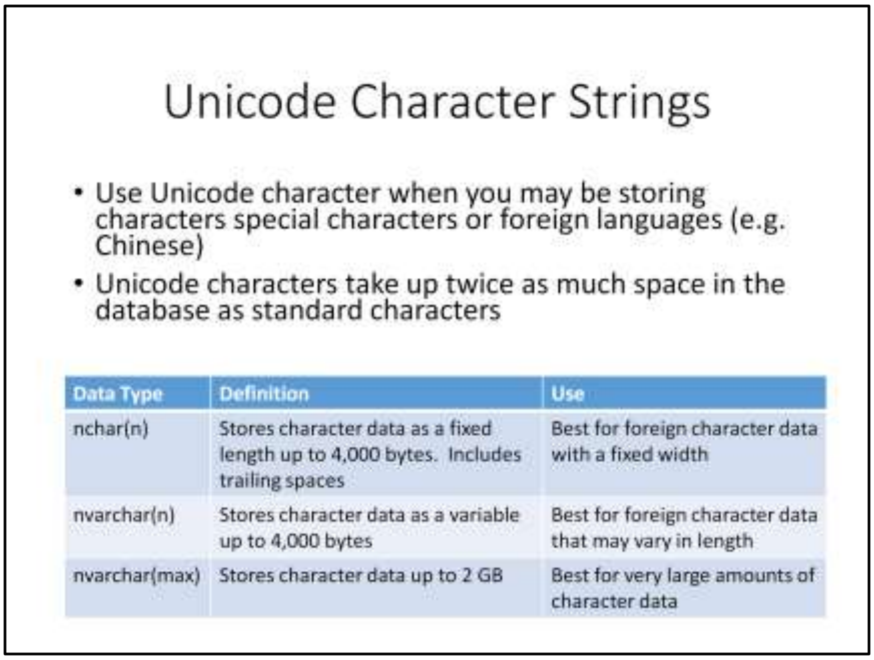


Character Strings

- Use character strings for storing standard alphanumeric data special characters
- This generally includes all the characters you will find on an American keyboard

Data Type	Definition	Use
Char(n)	Stores character data as a fixed length up to 8,000 bytes. Includes trailing spaces	Best for character data with a fixed width like zip codes
Varchar(n)	Stores character data as a variable up to 8,000 bytes	Best for character data that may vary in length
Varchar(max)	Stores character data up to 2 GB	Best for very large amounts of character data like an essay

Character strings are used for storing alphanumeric data. The char data type stores data as a fixed length. This means any string that doesn't reach the specified length will be padded with spaces. The varchar data type allows for variability in string length and doesn't add any space padding. Both the char and varchar datatypes need to have their maximum length written in the parenthesis after their name. The varchar(max) data type can be used to store very large strings up to 2 gigabytes. Only use varchar(max) when necessary because the data type is not efficient to query against. To summarize use **char** when the data length is consistent, use **varchar** when the data length may vary, and use **varchar(max)** when the data length might exceed 8,000 bytes.

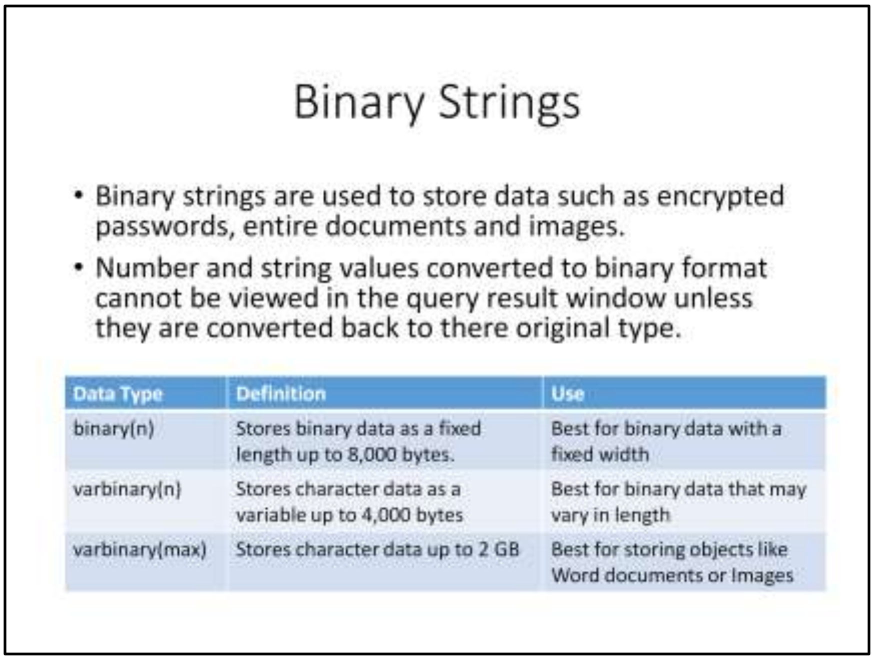


Unicode Character Strings

- Use Unicode character when you may be storing characters special characters or foreign languages (e.g. Chinese)
- Unicode characters take up twice as much space in the database as standard characters

Data Type	Definition	Use
nchar(n)	Stores character data as a fixed length up to 4,000 bytes. Includes trailing spaces	Best for foreign character data with a fixed width
nvarchar(n)	Stores character data as a variable up to 4,000 bytes	Best for foreign character data that may vary in length
nvarchar(max)	Stores character data up to 2 GB	Best for very large amounts of character data

Unicode character strings are used to store data that may include internationals characters or symbols. Nchar and nvarchar characters use twice the space in a SQL server that their char and varchar cousins do, but otherwise behave in the same manner. The nvarchar(max) also shares the same characteristics as varchar(max). A good rule of thumb is to use Unicode character strings if there is a chance you will be working in multiple languages.

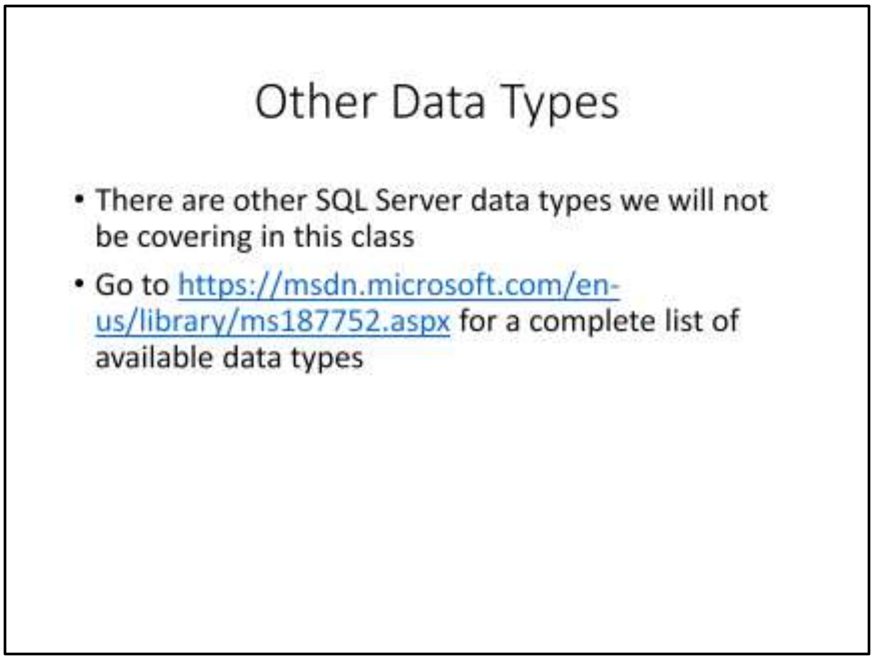


Binary Strings

- Binary strings are used to store data such as encrypted passwords, entire documents and images.
- Number and string values converted to binary format cannot be viewed in the query result window unless they are converted back to there original type.

Data Type	Definition	Use
binary(n)	Stores binary data as a fixed length up to 8,000 bytes.	Best for binary data with a fixed width
varbinary(n)	Stores character data as a variable up to 4,000 bytes	Best for binary data that may vary in length
varbinary(max)	Stores character data up to 2 GB	Best for storing objects like Word documents or Images

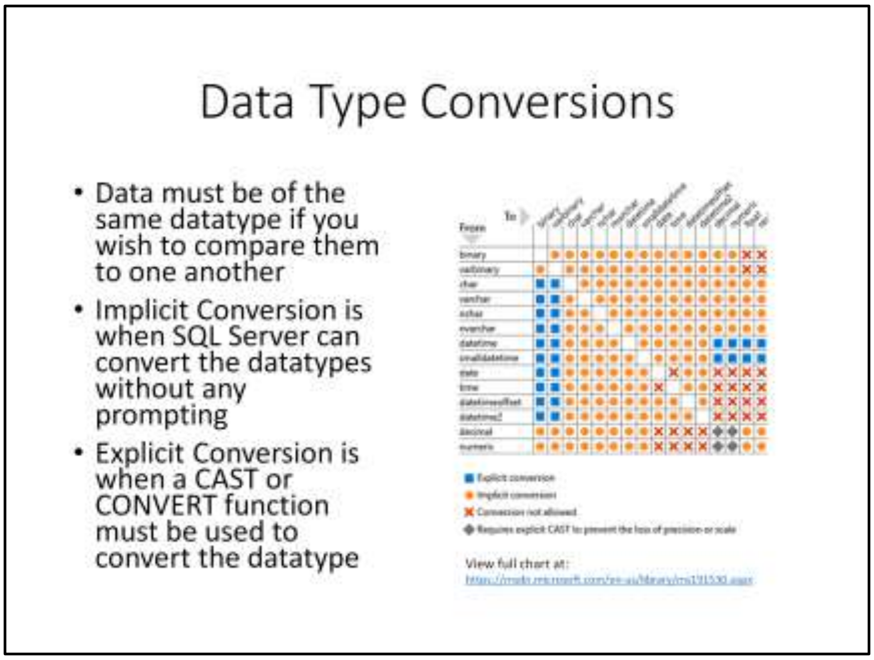
Binary strings are used to store data in a binary format. Images, pdfs, and other electronic documents are examples of what can be stored in binary. The binary, varbinary and varbinary(max) data types have the same qualities as their character cousins, but they only take binary input. It is possible to convert character data to binary, and this is often done with data that needs to be encrypted like passwords. Character data converted to binary cannot be read in the query results window. It needs to be converted back to character data in order to be readable.



Other Data Types

- There are other SQL Server data types we will not be covering in this class
- Go to https://msdn.microsoft.com/en-us/library/ms187752.aspx for a complete list of available data types

There are other data types that we will not be covering in this class. To view all possible data types, go to the MSDN page linked on this slide.



Data Type Conversions

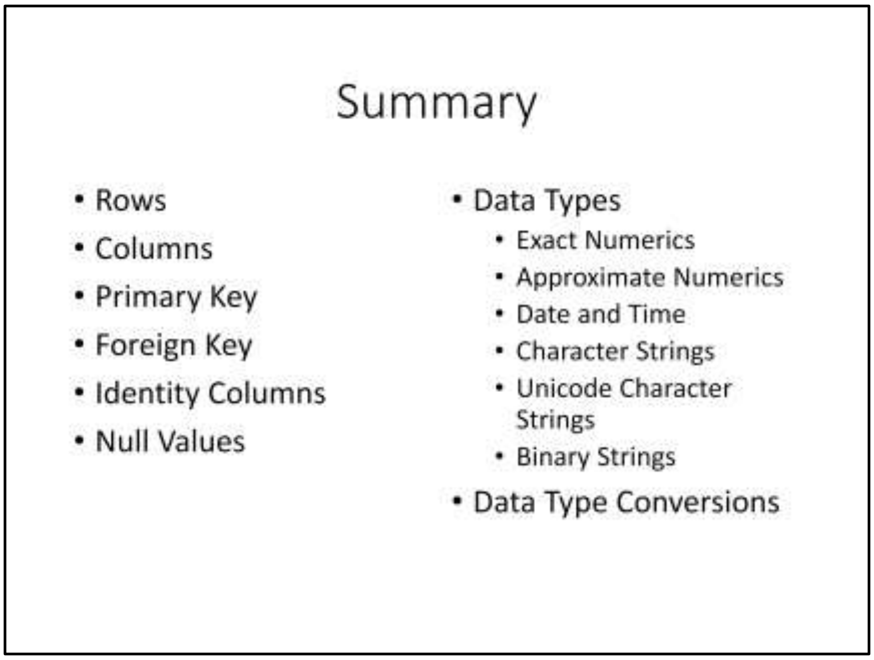
- Data must be of the same datatype if you wish to compare them to one another
- Implicit Conversion is when SQL Server can convert the datatypes without any prompting
- Explicit Conversion is when a CAST or CONVERT function must be used to convert the datatype

From \ To	binary	varbinary	char	varchar	nchar	nvarchar	datetime	smalldatetime	date	time	datetimeoffset	datetime2	decimal	numeric	float	real
binary		●	●	●	●	●	●	●	●	●	●	●	●	●	✗	✗
varbinary	●		●	●	●	●	●	●	●	●	●	●	●	●	✗	✗
char	■	■		●	●	●	●	●	●	●	●	●	●	●	●	●
varchar	■	■	●		●	●	●	●	●	●	●	●	●	●	●	●
nchar	■	■	●	●		●	●	●	●	●	●	●	●	●	●	●
nvarchar	■	■	●	●	●		●	●	●	●	●	●	●	●	●	●
datetime	■	■	●	●	●	●		●	●	●	●	●	■	■	■	■
smalldatetime	■	■	●	●	●	●	●		●	●	●	●	■	■	■	■
date	■	■	●	●	●	●	●	●		✗	●	●	✗	✗	✗	✗
time	■	■	●	●	●	●	●	●	✗		●	●	✗	✗	✗	✗
datetimeoffset	■	■	●	●	●	●	●	●	●	●		●	✗	✗	✗	✗
datetime2	■	■	●	●	●	●	●	●	●	●	●		✗	✗	✗	✗
decimal	●	●	●	●	●	●	●	●	✗	✗	✗	✗	◆	◆	●	●
numeric	●	●	●	●	●	●	●	●	✗	✗	✗	✗	◆	◆	●	●

■ Explicit conversion
● Implicit conversion
✗ Conversion not allowed
◆ Requires explicit CAST to prevent the loss of precision or scale

View full chart at:
https://msdn.microsoft.com/en-us/library/ms191530.aspx

It is important to know that columns need to be of the same datatype if you wish to compare them to one another. SQL Server automatically tries to convert some data types for you, such as integer to numeric. This is known as an implicit conversion. There are other data types where conversion is possible but SQL server cannot do it for you without additional prompting. This is know as explicit conversion. Finally there are data type combinations where conversions are not allowed. The slide shows part of the conversion chart for SQL Server. Go to the MSDN website for a complete list of conversion mappings.



Summary

- Rows
- Columns
- Primary Key
- Foreign Key
- Identity Columns
- Null Values
- Data Types
  - Exact Numerics
  - Approximate Numerics
  - Date and Time
  - Character Strings
  - Unicode Character Strings
  - Binary Strings
- Data Type Conversions

This concludes the presentation on table components.