Logo

Description automatically generated

Day 2

Relational Database Development

Azure Data Studio

Structured Query Language

Data Types

SQL Conventions

Primary Key

Internal Constraints

External Constraints

Dropping Constraints

Contents

[SQL Server Database engine 2](#_Toc177500160)

[SQL Server Management Studio 2](#_Toc177500161)

[Structured Query Language 4](#_Toc177500162)

[Create a Database 5](#_Toc177500163)

[Delete a Database 6](#_Toc177500164)

[Data Types 7](#_Toc177500165)

[Create a Table 7](#_Toc177500166)

[Delete a Table 8](#_Toc177500167)

[SQL Conventions 9](#_Toc177500168)

[Inserting Data 10](#_Toc177500169)

[Querying Data 10](#_Toc177500170)

[Filtering Data 12](#_Toc177500171)

[Primary Key 14](#_Toc177500172)

[Composite Primary Key 15](#_Toc177500173)

[Foreign Key 16](#_Toc177500174)

[Internal Constraints 18](#_Toc177500175)

[CHECK 18](#_Toc177500176)

[CHECK IN 19](#_Toc177500177)

[NULL or NOT NULL 20](#_Toc177500178)

[Default Values 21](#_Toc177500179)

[Auto-Increment 22](#_Toc177500180)

[External Constraints 22](#_Toc177500181)

[Adding Primary Keys 22](#_Toc177500182)

[Adding Foreign Keys 24](#_Toc177500183)

[Adding NOT NULL Constraint 25](#_Toc177500184)

[Enforcing Ranges 26](#_Toc177500185)

[Dropping Constraints 27](#_Toc177500186)

# SQL Server Database engine

The SQL Server Database Engine is a core component of Microsoft's SQL Server. It's designed to operate as a service that typically starts when Windows boots up. Once initiated, the database engine runs quietly in the background, ensuring that SQL Server is always ready to handle data requests.

At its heart, the engine is responsible for storing data securely and efficiently. Beyond just storage, the database engine processes SQL queries to interact with the database when requested.

## Azure Data Studio

While many database programmers traditionally use **SQL Server Management Studio (SSMS)**, on our setup we’ll be using **Azure Data Studio (ADS)**. It provides a modern graphical interface to connect to and manage the SQL Server Database Engine.

|  |  |
| --- | --- |
| A red triangle with a white exclamation mark  AI-generated content may be incorrect. | To continue with this course, you must have SQL Server (in Docker) and Azure Data Studio running. If you haven’t installed the software yet, please do so now and refer to the installation notes on the Learning Hub for assistance if needed. |

## Connect Azure Data Studio To SQL Server (Docker)

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Hands-on Practice:   1. **Open Azure Data Studio** 2. Click **New Connection**. 3. Connect to the SQL Server instance:  * **Server:** localhost,1433 * **Authentication:** SQL Login * **User name:** sa * **Password:** YourPassword * check **Remember password** * **Encrypt:** Mandatory * **Trust server certificate:** True * **Name:** *Docker SQL 2022*.  1. Click **Connect**. | |  | | --- | |  | | |

# Structured Query Language

Often pronounced as "sequel", SQL is an acronym for **Structured Query Language** . It’s the standard language for interacting with the SQL Server Database Engine.

## Create a Database

You can create a new database using the CREATE DATABASE statement, followed by your chosen database name, ending with a semicolon:

|  |
| --- |
| -- Creates a specific database  CREATE DATABASE Literature; |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Hands-on Practice:   1. On the top menu, click **New Query**. A new blank editor window will appear. 2. Paste the CREATE DATABASE statement above into the query window. 3. Click the Run button. 4. A message will appear in the Messages pane, indicating the success (or failure) of the operation as well as the time it took to complete the query. 5. To see your newly created database, expand the Databases section in the Object Explorer, right-click on Databases and choose Refresh. 6. Your new Literature database should now be visible. | |  | | --- | |  | | |

Quiz Questions 1, 2 and 3

## Delete a Database

To remove a database, employ the DROP DATABASE command, followed by the name of the database you wish to delete. Conclude the statement with a semicolon. The optional IF EXISTS clause ensures the command executes only if the specified database is present, preventing a potential “not exists” error.

|  |  |
| --- | --- |
| -- Removes a specific database if it exists  DROP DATABASE IF EXISTS Literature; | |
| A yellow bell with a white background  AI-generated content may be incorrect. | | Be cautious when using the DROP DATABASE command. It's a potent statement that permanently removes a database. | |

|  |
| --- |
| Hands-on Practice:  Delete the Database   * Copy the above DROP DATABASE command into ADS's query window. * Ensure that the previous CREATE DATABASE statement is removed, then click Run. * Refresh the database list in Object Explorer. * The Literature database should no longer be listed.   To continue with our exercises, recreate the Literature database as previously instructed.  Once your database is established, you must select it in the IDE;    Otherwise, any new tables will default to the Master database. |

## Data Types

SQL data types play a crucial role in both optimizing storage and ensuring data integrity. Below are some commonly used data types, although the full list is more extensive:

|  |  |  |
| --- | --- | --- |
| Type | | Description |
| BIGINT | Range: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 | |
| INT | Range: -2,147,483,648 to 2,147,483,647 | |
| SMALLINT | Range: -32768 to 32767 | |
| TINYINT | Range: 0 to 255 | |
| BIT | Accepts values: 0, 1, or NULL | |
| DECIMAL(m,d) | ***m* (length)** defaults to 18, if not specified. ***d* (decimals)**defaults to 0, if not specified | |
| NUMERIC | Functionally identical to DECIMAL | |
| MONEY | Range: -922,337,203,685,477.5808 to 922,337,203,685,477.5807 | |
| SMALLMONEY | Range: -214,748.3648 to 214,748.3647 | |
| DATE | Covers dates from January 1st, 0001 to December 31st, 9999 | |
| TIME | Represents hours, minutes, and seconds on a 24-hour clock | |
| CHAR(size) | Fixed length (max 8,000 characters). Best for consistent-length data | |
| VARCHAR(size) | Variable length (max size of 8,000 characters) | |

While there's no need to commit this list to memory, it's essential to choose efficient data types during table design. When in doubt, you can always refer to this list or check Microsoft's official SQL documentation for a comprehensive guide, here’s the link:

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15>

Quiz Question 4

## Create a Table

To define entities in your database, start with the CREATE TABLE statement followed by the desired table name. Next are parentheses and within them is a list of columns along with their data types. You conclude the statement with a semicolon:

SQL Keywords to create a new table.

|  |  |
| --- | --- |
| CREATE TABLE Book  (  pkBarcode VARCHAR(50) PRIMARY KEY,  title VARCHAR(50)  ); | primary key is declared here. |

column names are declared here. column data types are represented here

|  |  |  |
| --- | --- | --- |
| Now You Try:   |  |  | | --- | --- | | Enhance the Book table CREATE statement from the previous example by adding a price column. Ensure you set its data type to MONEY:  Execute your revised statement and once successfully completed, the new table will be created.  Navigate to the Object Explorer to inspect the table structure. If the changes aren't immediately visible, right-click and choose **Refresh** to update the displayed objects. |  | |

## Delete a Table

|  |  |  |
| --- | --- | --- |
| To delete a table and all its contained data, use the DROP statement: | |  | | --- | | DROP TABLE Book; | |

|  |  |  |
| --- | --- | --- |
| Like the create database statement used earlier, you can  include the optional IF EXISTS clause to ensure the command  executes only if the specified table is present. | |  | | --- | | DROP TABLE IF EXISTS Book; | |

When executing SQL statements, be careful to avoid unintended actions.

|  |  |
| --- | --- |
| A yellow bell with a white background  AI-generated content may be incorrect. | Here are some tips to avoid executing unintended SQL commands:   * Clear previous SQL statements from the query window before executing new ones. * Use the **New Query** button to open a fresh tab for crafting new statements. * To execute a specific block of code, simply highlight the desired code segment and then click the **Execute** button. |

## SQL Conventions

SQL is case-insensitive, which means it **does not** distinguish between uppercase and lowercase letters. However, to ensure clarity and maintainability, it's crucial to adopt consistent naming and styling conventions. By following these conventions, both you and others can swiftly discern the type and purpose of database objects.

In this SSD course, I want you to adhere to the following conventions:

|  |  |
| --- | --- |
| SQL Keywords | Always in UPPER CASE. |
| Database Names | Use PascalCase. e.g. DatabaseName. |
| Table Names | Follow PascalCase as well. e.g. TableName. |
| Column Names | Adopt camelCase. e.g. columnName. |
| Data Type Assignment | Be efficient when assigning data types. |
| Singular Notation | Ensure that both tables and columns use singular terms, not plural. |
| Special Characters | Refrain from using spaces or special characters in table or column names. |
| Primary Keys | Prefix with pk, followed by a PascalCase name. e.g. pkEmployeeId. |
| Foreign Keys | Prefix with fk, also followed by a PascalCase descriptor. e.g. fkEmployeeId. |
| Query Formatting | Keep your SQL queries tidy and organized. Proper indentation and formatting improve readability immensely. |

Quiz Question 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hands-on Practice:  Run the following commands to create a new database and table within it:   |  | | --- | | CREATE DATABASE Rubbermaid; |  |  |  | | --- | --- | | A red triangle with a white exclamation mark  AI-generated content may be incorrect. | In Azure Data Studio, make sure you select **Rubbermaid** from the **database dropdown at the top of the query editor** before running the next command. Otherwise, the new table will be created in whichever database is currently active (often *master*). |  |  | | --- | | CREATE TABLE LiquidContainer  (  pkLiquidType VARCHAR(30) PRIMARY KEY,  liters FLOAT  ) | |

## Inserting Data

The INSERT INTO keywords initiate the command to add new records to a specific table. These are followed by the table's name, then the VALUES keyword, and finally the data values for each column, enclosed in parentheses. The statement ends with a semicolon.

|  |  |  |
| --- | --- | --- |
| For example: | |  | | --- | | INSERT INTO LiquidContainer VALUES('canola oil', 50043); | |

Another way is to specify the columns you're inserting into. The next insert statement ensures that data gets added to the right places, even if given in a different order than the table's default.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Like this: | INSERT INTO LiquidContainer(liters, pkLiquidType) VALUES(43926, 'water'); | | | |
| A red triangle with a white exclamation mark  AI-generated content may be incorrect. | It's important to ensure the data order in the VALUES clause aligns with the specified column names. If columns aren't mentioned, as in the first insert command, then the data order must match the table's default column sequence. | |

Quiz Question 6

## Querying Data

To retrieve data from a database:

* Begin with the SELECT keyword.
* Then specify the columns you want to retrieve, separated by commas.

|  |  |  |
| --- | --- | --- |
| * Next use the FROM keyword followed by the table name. * Conclude with a semicolon to mark the end of the statement. | |  | | --- | | SELECT pkLiquidType  , liters  FROM LiquidContainer; | |

When retrieving all columns from a table, replace the column names with an asterisk.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Like this: | |  | | --- | | SELECT \*  FROM LiquidContainer; | |  |  | |

|  |  |
| --- | --- |
| Now You Try:  Run the following statements to delete then recreate the LiquidContainer table and add data to it.   |  | | --- | | -- Drop table if it exists.  DROP TABLE IF EXISTS LiquidContainer;  GO  -- Create table.  CREATE TABLE LiquidContainer  (  pkLiquidType VARCHAR(30) PRIMARY KEY,  liters FLOAT,  supplier VARCHAR(25)  );  -- Add data.  INSERT INTO LiquidContainer VALUES('canola oil', 50043, 'Western');  INSERT INTO LiquidContainer VALUES('petroleum' , 20000, 'Chevron');  INSERT INTO LiquidContainer VALUES('milk' , 45000, 'Dairyland'); | |

|  |  |  |
| --- | --- | --- |
| Now You Try - Continued:   1. Create a statement to select all columns and all rows from the *LiquidContainer* table. Executing your command should produce the following results:  |  | | --- | |  |  1. Create a statement to return only the liquidType and supplier columns from the LiquidContainer table. Executing your command should produce the following results:  |  | | --- | | A list of different types of liquids  AI-generated content may be incorrect. |   Quiz Questions 7 and 8 |

## Filtering Data

To selectively retrieve rows from a table based on certain conditions, you'll employ the WHERE clause. This clause filters rows from the resulting dataset based on the conditions you specify.

You can utilize various operators to form complex decision-making criteria. Below are some commonly used operators:

|  |  |  |
| --- | --- | --- |
| Operator | Example Syntax | Description |
| = | SELECT \*  FROM LiquidContainer  WHERE pkLiquidType = 'milk'; | Equality check.   |  | | --- | |  | |
| > | SELECT \*  FROM LiquidContainer  WHERE liters > 45000; | Values greater than the specified number.   |  | | --- | |  | |
| < | SELECT \*  FROM LiquidContainer  WHERE liters < 45000; | Values less than the specified number.   |  | | --- | |  | |
| >= | SELECT \*  FROM LiquidContainer  WHERE liters >= 45000; | Values greater than or equal to the specified number.   |  | | --- | |  | |
| <= | SELECT \*  FROM LiquidContainer  WHERE liters <= 45000; | Values less than or equal to the specified number.   |  | | --- | |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| AND | SELECT \*  FROM LiquidContainer  WHERE liters > 40000  AND supplier = 'Dairyland'; | Both conditions on either side of the AND must be true.   |  | | --- | |  | |
| OR | SELECT \*  FROM LiquidContainer  WHERE liters > 40000  OR supplier = 'Dairyland'; | Only one condition on either side of the OR must be true.   |  | | --- | | A white rectangular object with black text  AI-generated content may be incorrect. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Now You Try:   1. Write a query to retrieve liquid container records that have a canola oil liquid type. Executing your command should produce the following results:  |  | | --- | |  |  1. Write an SQL query to retrieve liquid container records that have less than 30000 litres. Executing your command should produce the following results:  |  | | --- | |  |  1. Write an SQL query to retrieve records where the supplier is either 'Western' or 'Dairyland'. Executing your command should produce the following results:  |  | | --- | |  |   Quiz Questions 9, 10, 11, 12 and 13 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Now You Try:  Run the following script:   |  | | --- | | -- Drop table if it exists.  DROP TABLE IF EXISTS Product;  GO  -- Create table.  CREATE TABLE Product  (  pkProductId INT PRIMARY KEY,  [name] VARCHAR(25),  Supplier VARCHAR(25),  unitPrice MONEY  );  -- Add data.  INSERT INTO Product VALUES(1, 'peanut butter', 'Golden Boy', 4.32);  INSERT INTO Product VALUES(2, 'peanut snacks', 'Golden Boy', 2.37);  INSERT INTO Product VALUES(3, 'fruit rollups', 'Kellog' , 4.00); |  1. Write a query to retrieve all records from the Product table. Executing your command will produce the following results:  |  | | --- | | A screenshot of a white grid with black text  Description automatically generated |  1. Write a query to only show products that are supplied by Golden Boy. Executing your command should produce the following results:  |  | | --- | | A screenshot of a phone  Description automatically generated |  1. Write a query to return products with either a *unitPrice* greater than $4.10 or is supplied by Kellogg. Executing your command should produce the following results:  |  | | --- | | A screenshot of a computer  Description automatically generated |   Quiz Question 14 |

## Primary Key

A primary key ensures that each record in a table is unique. In SQL, you can designate an attribute as a primary key in two main ways during table creation:

1. Inline: The PRIMARY KEY constraint is defined alongside the attribute itself.

|  |
| --- |
| CREATE TABLE Employee (  pkEmployeeId INTEGER PRIMARY KEY,  firstName VARCHAR(30),  lastName VARCHAR(30)  ); |

1. Separate Line: The PRIMARY KEY constraint is set separately at the end of the table definition.

|  |
| --- |
| CREATE TABLE Employee (  pkEmployeeId INTEGER,  firstName VARCHAR(30),  lastName VARCHAR(30),  PRIMARY KEY (pkEmployeeId)  ); |

## Composite Primary Key

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | CREATE TABLE Room (  pkBuildingName VARCHAR(30),  pkRoomNumber INT,  PRIMARY KEY (pkBuildingName, pkRoomNumber)  ); | A composite PRIMARY KEY consists of multiple columns used together as a unique key. |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Now You Try:  De-select the Literature from the active database then run the following script:   |  |  | | --- | --- | | |  | | --- | | -- Drop the Literature database  DROP DATABASE IF EXISTS Literature;  -- Create the Literature database  CREATE DATABASE Literature; | |  1. Set the active database to Literature then create a new table and name it Author.   The table should have the following columns:   * Author Id: INTEGER, Set this to the primary key. * First Name: VARCHAR(30) * Last Name: VARCHAR(30) * Phone: VARCHAR(15)  |  |  | | --- | --- | | |  | | --- | | Once created navigate to the Object Explorer to inspect the new table structure. | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Now You Try: Continued   1. Create a new table and name it publisher. The table should have the following columns:  * Publisher Id: INTEGER Set this to the primary key. * Publisher Name: VARCHAR(50)  |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | Once created navigate to the Object Explorer to inspect the new table structure. |  | |  1. Create a table named book edition which represents different editions of books in various languages. The table should have the following columns:  * Book Title: VARCHAR(50) Make book title and language a composite * language: VARCHAR(20) primary key for this table.  |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | Once created navigate to the Object Explorer to inspect the new table structure. |  | |   Quiz Question 15 |

## Foreign Key

The creation of a foreign key in a (child) table involves referencing an already established primary key from another (parent) table. This establishes a relationship between the two tables and enforces referential integrity, ensuring that records in the child table correspond to records in the parent table.

|  |  |
| --- | --- |
| CREATE TABLE Book (  bookId INT PRIMARY KEY,  title VARCHAR(255),  fkAuthorId INT,  FOREIGN KEY(fkauthorId) REFERENCES Author(pkAuthorId)  ); | The REFERENCES keyword specifies which table and column the foreign key is referencing. |

Define which field in the table is the foreign key.

|  |  |
| --- | --- |
| A red triangle with a white exclamation mark  AI-generated content may be incorrect. | When defining a foreign key in a table, the data type of the foreign key column must match the data type of the primary key it references in the parent table. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hands-on Practice:  De-select the Literature database then run the following script:   |  |  | | --- | --- | | |  | | --- | | -- Drop the Literature database  DROP DATABASE IF EXISTS Literature;  -- Create the Literature database  CREATE DATABASE Literature; | |   Set the active database to Literature then run the following script.   |  |  | | --- | --- | | |  | | --- | | -- Create the Authors table  CREATE TABLE Author (  pkAuthorID INT PRIMARY KEY,  [name] VARCHAR(100)  );  -- Create the Books table  CREATE TABLE Book (  pkBookID INT PRIMARY KEY,  title VARCHAR(255),  fkAuthorID INT,  FOREIGN KEY(fkAuthorID) REFERENCES Author(pkAuthorID)  ); | |   Once created navigate to the Object Explorer to inspect both table’s new structures.  In this example, the fkAuthorID field in the Book table is a foreign key that references the pkAuthorID primary key in the Author table. |

|  |  |
| --- | --- |
| A red triangle with a white exclamation mark  AI-generated content may be incorrect. | Ensure the table with the primary key (parent table) is declared and created before the table with the foreign key (child table). A foreign key must point to an existing primary key in another table. |

Quiz Questions 16 and 17

|  |  |  |
| --- | --- | --- |
| Now You Try:  De-select the DepartmentStore database then run the following script:   |  | | --- | | -- Drop the Literature database  DROP DATABASE IF EXISTS DepartmentStore;  -- Create the Literature database  CREATE DATABASE DepartmentStore; |   Set the active database to DepartmentStore then run the following script.   |  | | --- | | -- Drop tables if they exist  DROP TABLE IF EXISTS Employee;  DROP TABLE IF EXISTS Department;  GO  -- Declare the Department table  CREATE TABLE Department (  pkDeptId INTEGER,  deptName VARCHAR(20),  PRIMARY KEY (pkDeptId)  ); |   **Your Task:** Now that you have created a Department table in the DepartmentStore database. You need to create a new Employee table that will store details about each employee, including first name, second name and which department they belong to. To ensure data integrity, you want to use a foreign key in the Employee table that references the primary key of the Department table.  After creating the Employee table, check the Object Explorer to verify that the primary and foreign keys are added correctly:    Quiz Questions 18 and 19 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Now you Try:  You've successfully created the Department and Employee tables for the DepartmentStore database. Now do the following:   1. Create a new department record with the following details:  * pkDeptId: 1 * deptName: Electronics.  |  |  |  | | --- | --- | --- | | 1. Create a query to return all rows from the   Department table. The results should look  like this: | |  | | --- | |  | |  1. Create a new employee record with the following details:  * pkEmployeeId: 101 * firstName: John * lastName: Doe * fkDeptId: 1 (which means John Doe works in the Electronics department)  |  |  |  | | --- | --- | --- | | 1. Create a query to return all rows from the   Employee table. The results should look  like this: | |  | | --- | |  | | |

|  |  |
| --- | --- |
| A red triangle with a white exclamation mark  AI-generated content may be incorrect. | Before inserting a record into a child table, ensure that the corresponding parent record already exists. Otherwise, a foreign key constraint error will occur. |

Quiz Question 20

# Internal Constraints

Constraint limits the values that can be placed in a column. Using constraints helps maintain data integrity in your tables by ensuring that only valid data is stored.

## CHECK

By leveraging the CHECK constraint, you can ensure the data adheres to specified rules. For instance, to guarantee a price column always has values greater than 0, the CHECK constraint can be applied.

|  |  |
| --- | --- |
| Hands-on Practice:  Select the Literature database then run the following script:   |  | | --- | | -- Drop table if it exists.  DROP TABLE IF EXISTS Book;  CREATE TABLE Book  (  pkBarcode VARCHAR(50) PRIMARY KEY,  title VARCHAR(50),  price MONEY,  CHECK (price > 0)  ) | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hands-on Practice Continued:   |  |  |  | | --- | --- | --- | | After you've created the table, you can inspect its structure in the Object Explorer. Navigate to the Book table, expand its constraints section and the CHECK constraint will be revealed. | |  | | --- | |  | |   With the constraint in place, any attempt to insert data violating the rule will fail.   |  | | --- | | INSERT INTO Book  VALUES('mar 233', 'Judy Bloom Summer Days', -15.33); |   For example, executing the statement above will produce the following error because it tries to insert a negative price.   |  | | --- | | Msg 547, Level 16, State 0, Line 14 The INSERT statement conflicted with the CHECK constraint "CK\_\_Book\_\_price\_\_49C3F6B7". The conflict occurred in database "Literature", table "dbo.Book", column 'price'.  The statement has been terminated. | |

## CHECK IN

The combination of CHECK and IN constraints provides a way to ensure that column values come from a specific set of values. This is particularly useful when you have a predefined list of data points.

|  |  |
| --- | --- |
| Hands-on Practice:  Run the following script in the Department Store Database:   |  | | --- | | -- Drop table if it exists.  DROP TABLE IF EXISTS Bike;  -- Create table.  CREATE TABLE Bike  (  pkBikeID INT PRIMARY KEY,  bikeType VARCHAR(50) CHECK (bikeType IN ( 'tandem'  , 'mens'  , 'womens'))  );  -- Add data.  INSERT INTO Bike VALUES (1, 'tandem');  INSERT INTO Bike VALUES (2, 'womens'); | |

|  |  |
| --- | --- |
| Now You Try:  With the constraint in place, any attempt to insert a bikeType other than 'tandem', 'mens' and 'womens' will fail.  **Your Task:** Write an SQL statement that inserts a new bike into the Bike table. Ensure the data you're trying to insert follows the constraint rule set on the bikeType column.  If your insert statement is successful without any constraint violation errors, you will see a message like this:   |  | | --- | | (1 row affected)  Completion time: 2023-09-04T00:03:18.1637005-07:00 | |

Quiz Question 21

## NULL or NOT NULL

NULL indicates that a column can have a missing value. In contrast, the NOT NULL constraint ensures a column always contains a value, enhancing data integrity.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hands-on Practice:  Set the active database to DepartmentStore then run the following script:   |  |  | | --- | --- | | DROP TABLE IF EXISTS Employee;  CREATE TABLE Employee (  pkEmployeeID INTEGER PRIMARY KEY,  firstName VARCHAR(20) NOT NULL,  middleName VARCHAR(20) NULL,  lastName VARCHAR(20) NOT NULL,  ); | NOT NULL mandates that a value must always be provided for that column.  NULL signifies that a column's value can be absent. |   Above, a table named Employee is defined. Both firstName and lastName are compulsory fields (NOT NULL), while middleName is optional (NULL).  Trying to insert a record without providing values for the NOT NULL columns will raise an error.   |  |  | | --- | --- | | INSERT INTO Employee(pkEmployeeID, lastName)  VALUES(1, 'jones'); | Run this insert statement! |   Executing the statement above will produce the following error because the firstName column, which is mandatory, is omitted:   |  | | --- | | Msg 515, Level 16, State 2, Line 9 Cannot insert the value NULL into column 'firstName', table 'DepartmentStore.dbo.Employee'; column does not allow nulls. INSERT fails.  The statement has been terminated. | |

|  |
| --- |
| Now You Try:  **Your Task:** I would like you to insert two records into the Employee table.   1. The first record should provide values for the NOT NULL columns only and omit the optional NULL column. Achieve this by inserting a record for ‘Mary Jones’. 2. The second statements should provide values for every column, including the middleName. Achieve this by inserting a record for ‘Tom Jimmy Jones’.   **Tip:** To save time build on the INSERT INTO Employee() command above.  Quiz Question 22 |

## Default Values

You can set default values for table columns. So, if a record is inserted without a value for such a column, the default value will automatically be applied.

|  |  |  |
| --- | --- | --- |
| Hands-on Practice:  Run the script below:   |  |  | | --- | --- | | -- Drop the Guest table if it already exists  DROP TABLE IF EXISTS Guest;  -- Create the Guest table with a default  -- value for the 'citizenship' column  CREATE TABLE Guest (  pkGuestID INTEGER PRIMARY KEY,  firstName VARCHAR(20) NOT NULL,  lastName VARCHAR(30) NOT NULL,  citizenship VARCHAR(20) DEFAULT 'Canada'  );  -- Insert records into the Guest table  INSERT INTO Guest(pkGuestID, firstName, lastName)  VALUES(1, 'Mary', 'Jones');  INSERT INTO Guest  VALUES(2, 'Fred', 'Chow', 'Chinese'); | The default citizenship of 'Canada' is assigned when none is specified. | |

Quiz Questions 23 and 24

## Auto-Increment

You can automate the generation of primary key values using the IDENTITY property. This is especially useful when the primary key is an integer.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hands-on Practice:  Run the script below:   |  |  | | --- | --- | | -- Drop the Guest table if it already exists  DROP TABLE IF EXISTS Guest;  -- Create the Guest table with a starting value and  -- an increment value for the primary key.  CREATE TABLE Guest (  pkGuestID INTEGER PRIMARY KEY IDENTITY (1000,3),  firstName VARCHAR(20) NOT NULL,  lastName VARCHAR(30) NOT NULL  ); | The IDENTITY property allows you to set both a starting value and an increment value. |   Now execute the insert statements below:   |  |  | | --- | --- | | -- Insert records into the Guest table  INSERT INTO Guest(firstName, lastName)  VALUES('Mary', 'Jones');  INSERT INTO Guest(firstName, lastName)  VALUES('Jane', 'Halverson'); | The pkGuestID values are omitted from the INSERT statement. |   Executing a simple SELECT statement on the Guest table will show the Ids created by SQL Server.   |  |  | | --- | --- | |  | The pkGuestID values are assigned automatically, according to the IDENTITY property (1000,3). | |

Quiz Question 25

# External Constraints

External constraints ensure data integrity by enforcing unique keys and specific data ranges. Unlike internal constraints, which are defined within the table's CREATE statement, external constraints are added to existing tables using the ALTER TABLE command.

## Adding Primary Keys

In cases where a table has been created without a primary key, it's possible to define one later using an external constraint.

When you define a **primary key** on a table, the database checks that all rows follow the rules:

* Values must be **unique** (no duplicates).
* Values must be **not null** (every row needs a key).

If the data already meets these rules, the primary key is applied with no changes.

If the data violates these rules, you will need to **alter rows** before the primary key can be applied.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hands-on Practice:  Run the script below to set up the next exercise:   |  |  | | --- | --- | | -- Check if the Employee table exists and drop it if it does.  DROP TABLE IF EXISTS Employee;  GO  -- Create the Employee table.  CREATE TABLE Employee (  employeeID INT IDENTITY(1000,5),  firstName VARCHAR(25),  lastName VARCHAR(25)  );  -- Insert a sample record.  INSERT INTO Employee  VALUES('bob', 'jones'); |  |  |  |  |  |  | | --- | --- | --- | --- | | Navigate to the Employee table in the Object Explorer. Right click on the table and select Refresh. Then expand the Columns directory. As indicated to the right, there are no keys currently on the Employee table. | |  |  | | --- | --- | |  |  | |   Now run the ALTER statement below:   |  |  | | --- | --- | | -- Add a primary key constraint to the employeeID column.  ALTER TABLE Employee  ADD CONSTRAINT employeePK PRIMARY KEY(employeeID); |  |  |  |  |  |  | | --- | --- | --- | --- | | Navigate to the Employee table in the Object Explorer again. Right click on the Employee table and select Refresh. This time expanding the Columns directory will show the primary key is recognized. | |  |  | | --- | --- | |  |  | |   Returning rows from the Employee table shows no data has been lost.   |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | SELECT \* FROM Employee; | | |  | | --- | |  | | |

Quiz Question 26

|  |  |  |
| --- | --- | --- |
| Hands-on Practice:  Set the active database to DepartmentStore then run the script below to set up the next exercise.   |  |  | | --- | --- | | -- Drop the child table first, followed by the parent.  DROP TABLE IF EXISTS Employee;  DROP TABLE IF EXISTS Department;  GO  -- Create the parent table (Department) first, as  -- it will be referenced by the child table (Employee).  CREATE TABLE Department (  pkDeptId INTEGER IDENTITY(1,1) PRIMARY KEY,  deptName VARCHAR(20)  );  CREATE TABLE Employee (  pkEmployeeId INTEGER IDENTITY(1,1) PRIMARY KEY,  firstName VARCHAR(30),  lastName VARCHAR(30),  deptId INTEGER  );  -- Inserting a record into the Department table.  INSERT INTO Department (deptName)  VALUES ('Human Resources');  -- Inserting a record into the Employee.  INSERT INTO Employee (firstName, lastName, deptId)  VALUES ('John', 'Doe', 1); |  | |

## Adding Foreign Keys

You can introduce a foreign key constraint to an existing table, linking it to a primary key in another table. This enforces referential integrity by ensuring relationships between tables are maintained.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hands-on Practice:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | Refresh the Employee table in the Object Explorer. Then expand the Columns directory.  As indicated to the right, there is no foreign key currently assigned to the Employee table. | |  |  | | --- | --- | |  |  | | | |

Implementing a foreign key is akin to adding a primary key, but there are specific rules:

* The parent table must already exist.
* The foreign key should map to a primary key in the parent table.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hands-on Practice:  Run the ALTER statement below to create a new foreign key constraint on the Employee table.  Define which field in the Employee table is the foreign key.   |  |  | | --- | --- | | ALTER TABLE Employee  ADD CONSTRAINT employeeFK FOREIGN KEY(deptId) REFERENCES Department(pkDeptId); |  |   The REFERENCES keyword specifies which table and column the foreign key is referencing.   |  |  |  |  | | --- | --- | --- | --- | | Refresh the Employee table again in the Object Explorer. Then expand the Columns directory.  This time expanding the Columns directory will show the foreign key is recognized. | |  |  | | --- | --- | |  |  | |   Returning rows from the Employee table shows no data has been lost.   |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | SELECT \* FROM Employee; | | |  | | --- | |  | | |

Quiz Questions 27 and 28

## Adding NOT NULL Constraint

The NOT NULL constraint ensures a column always contains a value. If you try to insert or update a record setting this column to NULL, the database will produce an error.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | Currently, the Employee table permits the firstName and lastName columns to have NULL values. | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hands-on Practice:  Execute the script below to modify the Employee table. This alteration ensures both the firstName and lastName columns cannot store null values.   |  |  | | --- | --- | | -- Adding NOT NULL constraint to fName column  ALTER TABLE Employee  ALTER COLUMN firstName VARCHAR(30) NOT NULL;  -- Adding NOT NULL constraint to lName column  ALTER TABLE Employee  ALTER COLUMN lastName VARCHAR(30) NOT NULL; |  |  |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | |  | After applying these changes, the Employee table will mandate that both firstName and lastName have values and cannot be left null. | | |

## Enforcing Ranges

After table creation, there might be situations where you need to enforce a specific range of values for an attribute.

|  |  |  |  |
| --- | --- | --- | --- |
| Hands-on Practice:  Delete the College database if it exists, then create it.   |  |  | | --- | --- | | |  | | --- | | -- Drop the College database  DROP DATABASE IF EXISTS College;  -- Create the College database  CREATE DATABASE College; | |   Set the active database to College then run the script below to set up the next exercise.   |  | | --- | | CREATE TABLE CourseGrade (  courseCode VARCHAR(8),  studentID INT,  grade INTEGER  PRIMARY KEY (courseCode, studentID),  ); | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hands-on Practice: Continued  To ensure that all grades entered are positive, apply a constraint to the grade column:   |  | | --- | | ALTER TABLE CourseGrade  ADD CONSTRAINT gradeMinimum CHECK(grade> 0); |  |  |  |  |  | | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | Refresh the CourseGrade table in the Object Explorer. Then expand the Constraints directory.  As indicated to the right, there is a gradeMinimum constraint currently assigned to the CourseGrade table. | |  | | --- | |  | | | |

|  |  |
| --- | --- |
| Now You Try:  **Your Task:** Write an SQL statement that inserts a grade into the CourseGrade table. Ensure the data you're trying to insert follows the constraint rule set on the gradeMinimum column.  If your insert statement is successful without any constraint violation errors, you will see a message like this:   |  | | --- | | (1 row affected)  Completion time: 2023-09-04T00:03:18.1637005-07:00 | |

## Dropping Constraints

A previously added range constraint can be deleted using the DROP CONSTRAINT statement. The following example shows you how to remove the recently added gradeMinimum constraint.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hands-on Practice:  Run the DROP CONSTRAINT statement below to remove the gradeMinimum constraint on the CourseGrade table.   |  | | --- | | ALTER TABLE CourseGrade DROP CONSTRAINT gradeMinimum; |  |  |  |  |  | | --- | --- | --- | --- | | Refresh the CourseGrade table again in the Object Explorer. Then expand the Constraints directory.  As indicated to the right, there are  no constraints currently assigned to the CourseGrade table. | |  |  | | --- | --- | |  |  | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Now You Try:  Select the DepartmentStore database. Write a command to remove the previously added foreign key constraint on the Employee table.  If your drop constraint is successful, you will see a message like this:   |  | | --- | | Commands completed successfully.  Completion time: 2023-09-04T21:05:59.1208982-07:00 |  |  |  |  |  | | --- | --- | --- | --- | | Refresh the Employee table again in the Object Explorer. Then expand the Columns directory.  This time expanding the Columns directory will show the foreign key is no longer there. | |  |  | | --- | --- | |  |  | |   Quiz Questions 29 and 30 |