A photograph of a stack of books. In the foreground, the spines of several books are visible, showing various colors like blue, orange, and red. A white rectangular overlay is positioned in the upper right quadrant. Inside this overlay, the letters "LSBU" are printed in a large, bold, dark gray sans-serif font. To the left of "LSBU", the text "EST 1892" is printed vertically in a smaller, dark gray font.

EST 1892

**Module Title:** Big Data and Database Systems

**Course Name:** Computer Science Top-Up

**Module Code:**

**Student ID:**

## Table of Contents

<b>1. Abstract .....</b>	1
<b>2. Introduction .....</b>	2
<b>3. Entity Relationship Diagram .....</b>	3
3.1 Objectives of the model.....	3
3.2 Key entities and relationships .....	3
<b>5. Transforming design to implementation .....</b>	9
<b>Query 4. a .....</b>	31
<b>Query 4. b.....</b>	32
<b>Query 4. c .....</b>	33
<b>Query 4. d.....</b>	34
<b>Query 4. e .....</b>	37
<b>7. Data Visualization .....</b>	40
<b>8. Video Demonstration .....</b>	41
<b>9. References .....</b>	42
<b>10. Appendix .....</b>	43

Figure 1.ERD .....	5
Figure 2. ERD by Groups .....	6
Figure 3. Create Client .....	10
Figure 4.Insert Client Values.....	10
Figure 5.Create Building .....	11
Figure 6. Insert Building Values .....	11
Figure 7. Create ClientBuilding .....	12
Figure 8. Insert ClientBuilding Values.....	12
Figure 9. Create Sensor .....	13
Figure 10. Insert Sensor Values .....	13
Figure 11.Create Controller .....	14
Figure 12.Insert Controller Values.....	14
Figure 13. Create SpecialDevice .....	15
Figure 14. Insert SpecialDevice Values.....	15
Figure 15. Create Design.....	16
Figure 16. Insert Design Values .....	16
Figure 17. Create DesignComponent .....	17
Figure 18. Insert DesignComponent Values .....	17
Figure 19. Create SensorControllerCompatibility .....	18
Figure 20. Insert SensorControllerCompatibility Values .....	18
Figure 21. Create Supplier .....	19
Figure 22. Insert Supplier Values.....	19
Figure 23. Create StockItem .....	20
Figure 24. Insert Stockitem Values .....	20
Figure 25. Create StockDevice .....	21
Figure 26. Insert StockDevice Values.....	21
Figure 27. Create StockCheckLog .....	22
Figure 28. Insert StockCheckLog Values .....	22
Figure 29. Create Staff .....	23
Figure 30. Insert Staff Values.....	23
Figure 31. Create InstalationTeam .....	24
Figure 32. Insert InstalationTeam Values .....	24
Figure 33. Create TeamMember .....	25
Figure 34. Insert TeamMember Values .....	25
Figure 35. Create ComponentInstallationLog.....	26
Figure 36. Insert ComponentInstallationLog Values .....	26
Figure 37. create Invoice .....	27
Figure 38. Insert Invoice Values.....	27
Figure 39. Create PaymentMethod .....	28
Figure 40. Insert PaymentMethod Values.....	28
Figure 41. Create Payment .....	29
Figure 42. Insert Payment Values.....	29
Figure 43. ClientBuilding Correction.....	30
Figure 44. Query 4. A .....	31
Figure 45. Query 4. B issue .....	32
Figure 46. Query 4.b corrected .....	33
Figure 47. Query 4.c .....	34
Figure 48. Trigger Creation.....	35

Figure 49. Trigger testing 1.....	35
Figure 50. Trigger testing 2.....	36
Figure 51. Trying Trigger.....	37
Figure 52. Procedure Creation .....	38
Figure 53. Procedure result .....	38
Figure 54. Dashboard .....	40

## 1. Abstract

This work focuses on the design of a database system for LSBU Smart Home, which automates the management of clients, buildings, and IoT sensors. Through the implementation of queries and stored procedures, as shown in the code figures, the system allows for efficient management of device inventory and the assignment of specialized personnel for installations. The trg\_ProtectDoubleBooking trigger prevents double staffing, and queries allow for generating reports on installations performed within a specific period, as seen in the queries for details of installations performed in a given month. The billing process is also facilitated by incorporating new payment methods. This system optimizes LSBU Smart Home's operations, ensuring resource availability and the proper integration of devices in each installation.

## 2. Introduction

In the new era of technology, the continuous development of information systems is essential for businesses to keep growing, especially in sectors like the Internet of Things (IoT). Advances in smart sensors have made it possible for homes to improve security, efficiency, and comfort for people. In this context, LSBU Smart Home specializes in the design, maintenance, and automation of smart home systems by implementing the most advanced IoT sensors.

The aim of this assignment is to design a prototype information system for LSBU Smart Home to optimize designs, teamwork, client management, inventory management, and control of IoT components. It also focuses on collecting data from installed IoT sensors in order to support better real-time decision-making.

A comprehensive database design has been proposed to efficiently store and manage data, linking it directly to the operational processes of LSBU Smart Home, as described in the Case Study. Some of these implementations require the integration of IoT sensors with controllers, strong communication with a central database, and the potential use of cloud platforms to support flexible payment methods such as PayPal and Google Pay.

## 3. Entity Relationship Diagram

### 3.1 Objectives of the model

The objective of this model is to build an operational system for LSBU Smart Home that includes managing clients, buildings, personalized IoT sensor designs, installations, inventory control with multiple suppliers, staff allocation, and the collection and analysis of smart home data.

### 3.2 Key entities and relationships

The initial entities considered in the initiation of this project were the following:

- Client
- Building
- Design
- DesignComponent
- Sensor
- Controller
- Staff
- InstallationTeam
- Supplier
- StockItem
- StockDevice
- SensorReading
- CentralSensorLog
- Invoice
- Payment

These entities served as the basis for developing the LSBU Smart Home database system, based on the functional and business requirements provided in Case Study C. However, while working on the physical database using SQL Server, it was necessary to adjust and expand the relational model to better reflect real-world scenarios.

As a result, the following tables were added:

- *ClientBuilding*, to allow multiple clients to be linked to the same building
- *TeamMember*, to enable flexible staff assignment across multiple teams
- *SensorControllerCompatibility*, to register compatibility between specific sensors and controllers

- *StockCheckLog*, to track inventory changes and maintain stock accuracy over time

These changes are observed in *Figure 1*, which shows the updated Entity Relationship Diagram (ERD) that includes 20 normalized tables. The ERD follows the IE Crow's Foot notation, used to represent *one-to-one*, *one-to-many*, and *many-to-many* relationships. All primary and foreign keys were implemented and verified through constraints in SQL Server, ensuring that the structure is relationally sound and accurately aligned with the business model (Dybka, 2016).

During the creation of the LSBU Smart Home database, I made several changes to the relational model in order to meet the requirements set in Case Study C. Initially, the model only included a few basic entities like *Client*, *Building*, *Sensor*, *Controller*, and *Invoice*. But as the system developed and was tested through SQL scripts, it became clear that more tables were needed to support the necessary relationships and business rules.

- Since several clients can share a single building, I added the *ClientBuilding* table
- Each building can have its own unique design, whether implemented by LSBU or not
- Sensor and controller compatibility needed to be recorded, so I added *SensorControllerCompatibility*
- Installation teams required flexible staff management, which led to the creation of *TeamMember*
- Periodic sensor readings needed to be sent securely to LSBU's central database, which required *CentralSensorLog*
- A proper inventory control system with support for multiple suppliers needed to be implemented using *StockItem*, *StockDevice*, and *StockCheckLog*

Based on these needs, I modified the *ClientBuilding* table to manage shared buildings, created *DesignComponent* to capture specific components used within each design, and added *TeamMember* to allow flexible team management for installations. The *SensorControllerCompatibility* table was added to track which controllers work with specific sensors, and *CentralSensorLog* was introduced to transmit sensor data to the LSBU data centre securely.

Finally, I included *Invoice* and *Payment* tables to fulfil the requirement of allowing future integration of multiple payment methods, including PayPal and Google Pay.

Once again in *Figure 1*, the final Entity Relationship Diagram (ERD) shows the complete data model containing 20 normalized tables, all of which follow Third Normal Form (3NF) and, in some cases, Boyce-Codd Normal Form (BCNF) (Visual Paradigm, 2019). The ERD demonstrates proper use of primary and foreign keys, and the data integrity has been enforced through SQL constraints. The tables have also been populated with realistic and logically connected test data that match the structure and logic described in the Case Study.

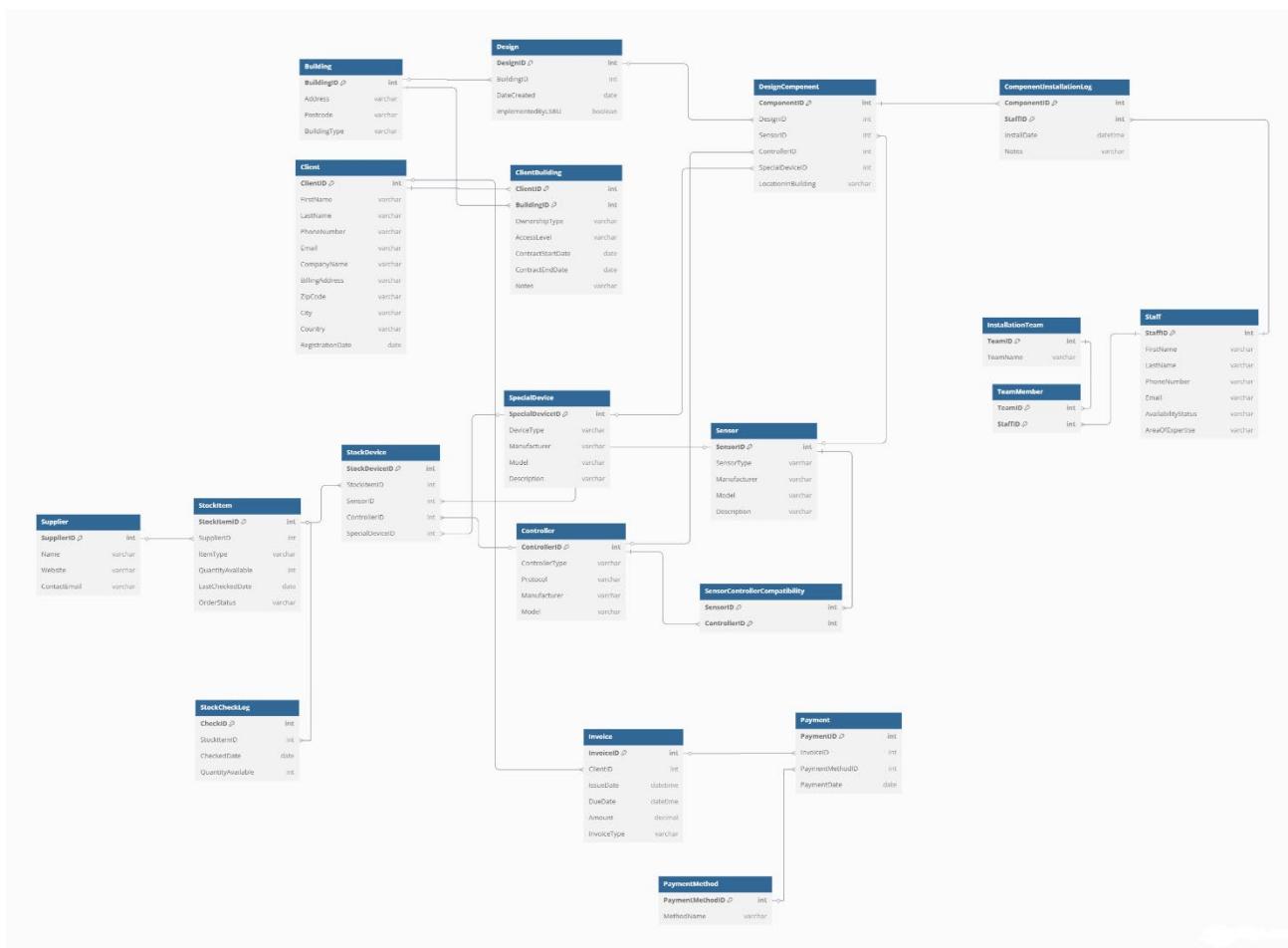


Figure 1.ERD

As well, *Figure 2*, shows the ERD table with the relation between the groups:

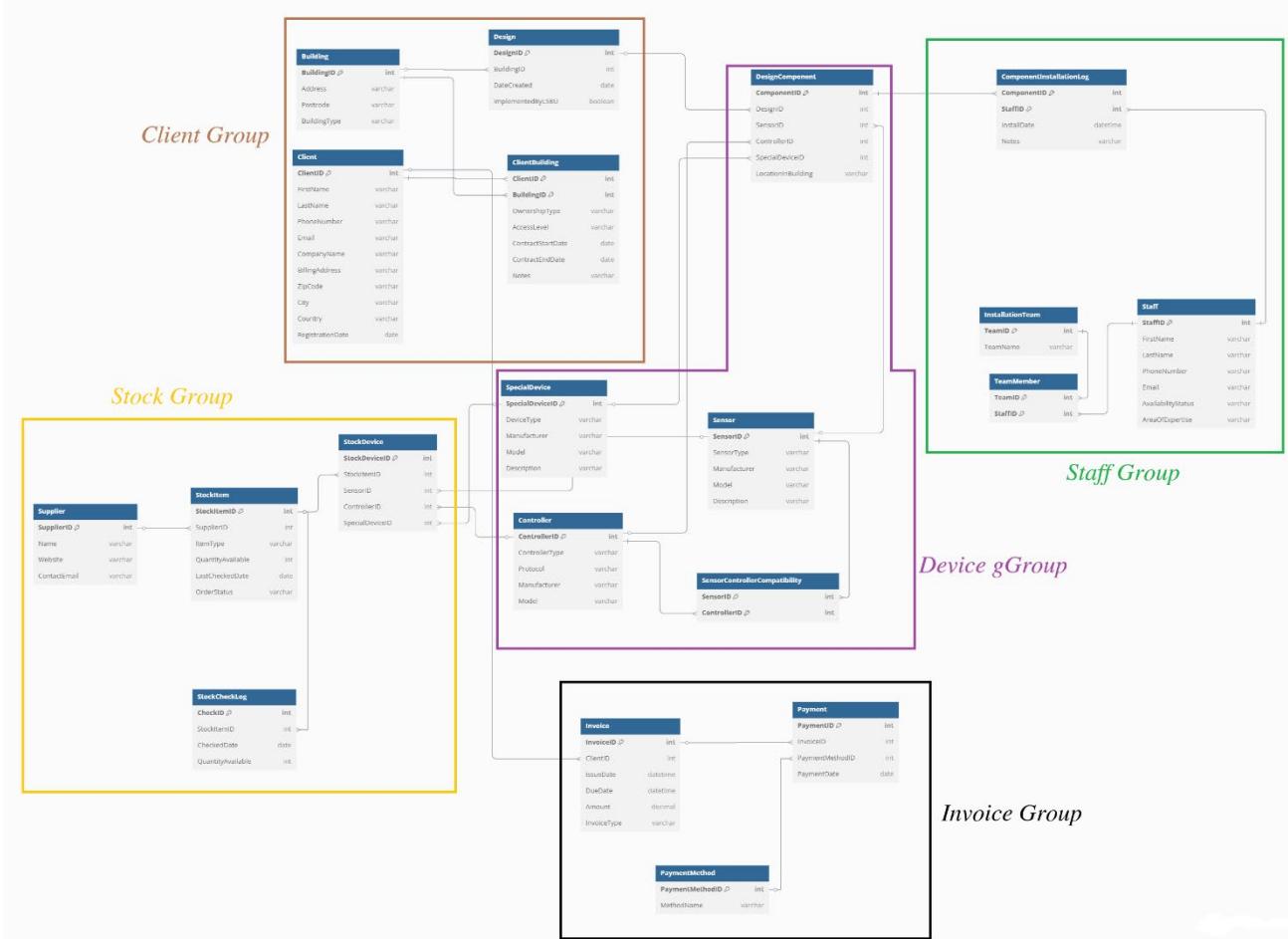


Figure 2. ERD by Groups

## 4. Functional Dependency statements

Functional dependency describes how a column in a table relates to other columns. It means that one attribute's value can determine another's value. This concept is important to maintain integrity and consistency in a database. Functional dependencies are also a key part of database normalization, which helps avoid data duplication and maintain accuracy (Staff, 2024).

Normalization is a design process used to reduce redundancy and prevent database anomalies. It ensures that data is logically organized across related tables. The process is guided by several normal forms (Microsoft, 2024), each with specific rules.

### **First Normal Form (1NF):**

All attributes must be atomic. Each column contains only one value and a primary key uniquely identifies each row. This helps structure data clearly and eliminates duplicate values in columns (Microsoft, 2024).

### **Second Normal Form (2NF):**

Once a table is in 1NF, 2NF checks that every non-key attribute depends on the entire primary key. This prevents partial dependencies, which occur when a non-key attribute depends on only part of a composite key. In 2NF, all attributes describe the entity represented by the full primary key (Microsoft, 2024).

### **Third Normal Form (3NF):**

After a table reaches 2NF, the goal is to optimize its structure. In this stage, transitive dependencies must be removed. This means no non-key attribute should depend on another non-key attribute. Every non-key value should depend directly on the primary key. This helps maintain a clean, logical structure without unnecessary or indirect relationships (Microsoft, 2024).

In *Table 1*, the 20 tables that make up the relational model are presented, along with their corresponding attributes and functional dependencies. As shown, each table has a clearly defined primary key, and all non-key attributes depend solely on that primary key, which confirms compliance with the Second Normal Form (2NF).

For example:  $\text{InvoiceID} \rightarrow \text{ClientID}$ ,  $\text{IssueDate}$ ,  $\text{DueDate}$ ,  $\text{Amount}$ ,  $\text{InvoiceType}$  in the *Invoice* table.

Additionally, it can be confirmed that there are no transitive dependencies since no non-key attribute determines another non-key attribute within the same table. All attributes depend directly and exclusively on their respective primary key, as seen in the *Sensor* table:

$\text{SensorID} \rightarrow \text{SensorType}$ ,  $\text{Manufacturer}$ ,  $\text{Model}$ ,  $\text{Description}$ .

Finally, it is important to highlight that all functional dependencies originate from primary keys to non-key attributes, fulfilling the requirements of the Third Normal Form (3NF). The relationships have been correctly normalized, and bridge tables such as *ClientBuilding* and *TeamMember* use composite keys properly to represent many-to-many relationships, reinforcing the integrity and consistency of the relational model design.

Table 1. Functional Dependency

TABLE	Functional Dependency
Client	ClientID → FirstName, LastName, PhoneNumber, Email, CompanyName, BillingAddress, ZipCode, City, Country, RegistrationDate
Building	BuildingID → Address, Postcode, BuildingType
ClientBuilding	(ClientID, BuildingID) → OwnershipType, AccessLevel, ContractStartDate, ContractEndDate, Notes
Sensor	SensorID → SensorType, Manufacturer, Model, Description
Controller	ControllerID → ControllerType, Protocol, Manufacturer, Model
SpecialDevice	SpecialDeviceID → DeviceType, Manufacturer, Model, Description
Design	DesignID → BuildingID, DateCreated, ImplementedByLSBU
DesignComponent	ComponentID → DesignID, SensorID, ControllerID, SpecialDeviceID, LocationInBuilding
SensorControllerCompatibility	(SensorID, ControllerID) → Compatibility
Supplier	SupplierID → Name, Website, ContactEmail
StockItem	StockItemID → SupplierID, ItemType, QuantityAvailable, LastCheckedDate, OrderStatus
StockDevice	StockDeviceID → StockItemID, SensorID, ControllerID, SpecialDeviceID
StockCheckLog	CheckID → StockItemID, CheckedDate, QuantityAvailable
Staff	StaffID → FirstName, LastName, PhoneNumber, Email, AvailabilityStatus, AreaOfExpertise
InstallationTeam	TeamID → TeamName
TeamMember	(TeamID, StaffID) → RoleInTeam
ComponentInstallationLog	(ComponentID, StaffID) → InstallDate, Notes
Invoice	InvoiceID → ClientID, IssueDate, DueDate, Amount, InvoiceType
PaymentMethod	PaymentMethodID → MethodName
Payment	PaymentID → InvoiceID, PaymentMethodID, PaymentDate

## 5. Transforming design to implementation

In this section, I will demonstrate through screenshots the creation of the 20 tables of this database, as well as the insertion of values into each table using Microsoft SQL Server.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'Smart\_Home\_LSBU' is selected. In the center pane, a script window titled 'LAPTOP-PSP6BBV6...\_LSBU - Diagram.1\*' contains the following T-SQL code:

```

-- Table: Client
CREATE TABLE Client (
    ClientID INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(100) NOT NULL,
    LastName NVARCHAR(100) NOT NULL,
    PhoneNumber NVARCHAR(20),
    Email NVARCHAR(255),
    CompanyName NVARCHAR(255),
    BillingAddress NVARCHAR(255),
    ZipCode NVARCHAR(10),
    City NVARCHAR(100),
    Country NVARCHAR(100),
    RegistrationDate DATE NOT NULL DEFAULT GETDATE()
);

SELECT * FROM Client;

-- Table: Building
CREATE TABLE Building (
    BuildingID INT PRIMARY KEY IDENTITY(1,1),
    Address NVARCHAR(255) NOT NULL,
    Postcode NVARCHAR(20),
    BuildingType NVARCHAR(100)
);

SELECT * FROM Building;

-- Table: ClientBuilding
CREATE TABLE ClientBuilding (
    ClientID INT NOT NULL,
    BuildingID INT NOT NULL,
    OwnershipType NVARCHAR(50),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID),
    FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID)
);

```

In the results pane, a table named 'PaymentMethod' is shown with the following data:

PaymentMethodID	MethodName
1	Credit Card
2	Debit Card
3	PayPal
4	Google Pay
5	Apple Pay

A status bar at the bottom indicates: 'Query executed successfully.' and 'LAPTOP-PSP6BBV6\SQLEXPRESS ... LAPTOP-PSP6BBV6\raul\_ ... Smart\_Home\_LSBU 00:00:00 5 rows'.

Figure 3. Create Client

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'Smart\_Home\_LSBU' is selected. In the center pane, a script window titled 'LAPTOP-PSP6BBV6...\_LSBU - Diagram.0\*' contains the following T-SQL code:

```

INSERT INTO Client (FirstName, LastName, PhoneNumber, Email, CompanyName, BillingAddress, ZipCode, City, Country, RegistrationDate)
VALUES
('John', 'Smith', '07400111222', 'john.smith@smithsol.co.uk', 'Smith Solutions', '221B Baker Street', 'NW1 6XE', 'London', 'UK', '2023-01-15'),
('Emily', 'Watson', '07500087654', 'emily.watson@watsonltd.uk', 'Watson Technologies', '14 Liverpool Street', 'EC2M 7PP', 'London', 'UK', '2023-02-18'),
('Raj', 'Patel', '07399443567', 'raj.patel@greenhome.co.uk', 'GreenHome Systems', '45 Victoria Road', 'W3 6BL', 'London', 'UK', '2022-11-20'),
('Aisha', 'Hassan', '07789992233', 'aisha.hassan@smartbuild.uk', 'SmartBuild Ltd', '3A Stratford Avenue', 'E15 4QS', 'London', 'UK', '2023-03-05'),
('Oliver', 'Brown', '07512345678', 'oliver.brown@brownenergy.co.uk', 'Brown Energy', '101 Camden High Street', 'NW1 7JN', 'London', 'UK', '2022-12-01'),
('Sophie', 'Turner', '07450987612', 'sophie.turner@techspace.uk', 'TechSpace Ltd', '7A Oxford Street', 'W1D 2LT', 'London', 'UK', '2023-01-25'),
('Mohammed', 'Ali', '07780123456', 'mohammed.ali@futureconnect.uk', 'Future Connect', '28 Clapham Road', 'SW9 0SG', 'London', 'UK', '2023-03-12'),
('Isabella', 'King', '07344321234', 'isabella.king@bulldsmart.uk', 'BullSmart Plc', '88 Kensington Road', 'W8 5NX', 'London', 'UK', '2023-01-18'),
('Thomas', 'Green', '07566778899', 'thomas.green@greenit.uk', 'GreenIT Solutions', '1E1 6PG', 'London', 'UK', '2022-10-30'),
('Charlotte', 'Wood', '07400099887', 'charlotte.wood@homeiq.co.uk', 'HomeIQ Ltd', '55 Brixton Hill', 'SW2 1HT', 'London', 'UK', '2023-02-14'),
('Jake', 'Reed', '07455678900', 'jake.reed@needtech.co.uk', 'NeedTech Ltd', '100 Whitechapel Road', 'E1 1JG', 'London', 'UK', '2023-01-08'),
('Lily', 'Bennett', '07300223344', 'lily.bennetts@smartzone.co.uk', 'SmartZone Services', '62 Notting Hill Gate', 'W11 3HT', 'London', 'UK', '2023-03-03'),
('Mason', 'Wright', '07577889922', 'mason.wright@urbanwave.co.uk', 'UrbanWave Ltd', '77 Tottenham Court Road', 'W1T 4JZ', 'London', 'UK', '2022-09-18'),
('Grace', 'Morgan', '07334556677', 'grace.morgan@connectedhomes.uk', 'Connected Homes Ltd', '4 Hackney Road', 'E2 7NX', 'London', 'UK', '2023-03-10'),
('Leo', 'Anderson', '07490011223', 'leo.anderson@echhaven.co.uk', 'TechHaven', '5 Croydon High Street', 'CR0 1QQ', 'London', 'UK', '2023-02-01'),
('Chloe', 'Robinson', '07778988811', 'chloe.robinson@solarity.uk', 'Solarity Energy', '39 King's Cross Road', 'WC1X 9LP', 'London', 'UK', '2023-03-09'),
('Noah', 'Scott', '07511223344', 'noah.scott@innovateliving.co.uk', 'Innovate Living', '15 Bermondsey Street', 'SE1 2BT', 'London', 'UK', '2022-11-25'),
('Ella', 'Hughes', '07393456789', 'ella.hughes@londoniot.uk', 'London IoT', '93 Peckham Rye', 'SE15 4JR', 'London', 'UK', '2023-01-30'),
('Oscar', 'Hall', '076000112345', 'oscar.hall@energyloop.co.uk', 'EnergyLoop Ltd', '21 Westbourne Grove', 'W1 4UA', 'London', 'UK', '2023-02-20'),
('Freya', 'Davies', '07403334455', 'freya.davies@nextgenhome.co.uk', 'NextGen Home Ltd', '6 Elephant Road', 'SE17 1LB', 'London', 'UK', '2023-03-06');

```

In the results pane, a table named 'Client' is shown with the following data:

ClientID	FirstName	LastName	PhoneNumber	Email	CompanyName	BillingAddress	ZipCode	City	Country	RegistrationDate
1	John	Smith	07400111222	john.smith@smithsol.co.uk	Smith Solutions	221B Baker Street	NW1 6XE	London	UK	2023-01-15
2	Emily	Watson	07500087654	emily.watson@watsonltd.uk	Watson Technologies	14 Liverpool Street	EC2M 7PP	London	UK	2023-02-18
3	Raj	Patel	07399443567	raj.patel@greenhome.co.uk	GreenHome Systems	45 Victoria Road	W3 6BL	London	UK	2022-11-20
4	Aisha	Hassan	07789992233	aisha.hassan@smartbuild.uk	SmartBuild Ltd	3A Stratford Avenue	E15 4QS	London	UK	2023-03-05
5	Oliver	Brown	07512345678	oliver.brown@brownenergy.co.uk	Brown Energy	101 Camden High Street	NW1 7JN	London	UK	2022-12-01
6	Sophie	Turner	07450987612	sophie.turner@techspace.uk	TechSpace Ltd	7A Oxford Street	W1D 2LT	London	UK	2023-01-25
7	Mohammed	Ali	07780123456	mohammed.ali@futureconnect.uk	Future Connect	28 Clapham Road	SW9 0SG	London	UK	2023-03-12
8	Isabella	King	07344321234	isabella.king@bulldsmart.uk	BullSmart Plc	88 Kensington Road	W8 5NX	London	UK	2023-01-18
9	Thomas	Green	07566778899	thomas.green@greenit.uk	GreenIT Solutions	1E1 6PG	London	UK	2022-10-30	
10	Charlotte	Wood	07400099887	charlotte.wood@homeiq.co.uk	HomeIQ Ltd	55 Brixton Hill	SW2 1HT	London	UK	2023-02-14
11	Jake	Reed	07455678900	jake.reed@needtech.co.uk	NeedTech Ltd	100 Whitechapel Road	E1 1JG	London	UK	2023-01-08
12	Lily	Bennett	07300223344	lily.bennetts@smartzone.co.uk	SmartZone Services	62 Notting Hill Gate	W11 3HT	London	UK	2023-03-03
13	Mason	Wright	07577889922	mason.wright@urbanwave.co.uk	UrbanWave Ltd	77 Tottenham Court Road	W1T 4JZ	London	UK	2022-09-18
14	Grace	Morgan	07334556677	grace.morgan@connectedhomes.uk	Connected Homes Ltd	4 Hackney Road	E2 7NX	London	UK	2023-03-10
15	Leo	Anderson	07490011223	leo.anderson@echhaven.co.uk	TechHaven	5 Croydon High Street	CR0 1QQ	London	UK	2023-02-01

A status bar at the bottom indicates: 'Query executed successfully.' and 'LAPTOP-PSP6BBV6\SQLEXPRESS ... LAPTOP-PSP6BBV6\raul\_ ... Smart\_Home\_LSBU 00:00:00 20 rows'.

Figure 4. Insert Client Values

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The central pane displays three CREATE TABLE statements:

```

--CREATE TABLE Building (
    BuildingID INT PRIMARY KEY IDENTITY(1,1),
    Address NVARCHAR(255) NOT NULL,
    Postcode NVARCHAR(20),
    BuildingType NVARCHAR(100)
);

SELECT * FROM Building;

-- Table: ClientBuilding
--CREATE TABLE ClientBuilding (
    ClientID INT NOT NULL,
    BuildingID INT NOT NULL,
    OwnershipType NVARCHAR(50),
    AccessLevel NVARCHAR(50),
    ContractStartDate DATE,
    ContractEndDate DATE,
    Notes NVARCHAR(255),
    PRIMARY KEY (ClientID, BuildingID),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID),
    FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID)
);

SELECT * FROM ClientBuilding;

-- Table: Sensor
--CREATE TABLE Sensor (
    SensorID INT PRIMARY KEY IDENTITY(1,1),
    SensorType NVARCHAR(100) NOT NULL,
    Manufacturer NVARCHAR(100),
    Model NVARCHAR(100),
    Description NVARCHAR(255)
);

```

The 'Results' tab shows the output of the SELECT statements. The first SELECT statement returns 100 rows of sample data for the Building table. The second SELECT statement returns 5 rows of sample data for the ClientBuilding table. The third SELECT statement returns 1 row of sample data for the Sensor table.

Figure 5. Create Building

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The central pane displays an INSERT INTO statement:

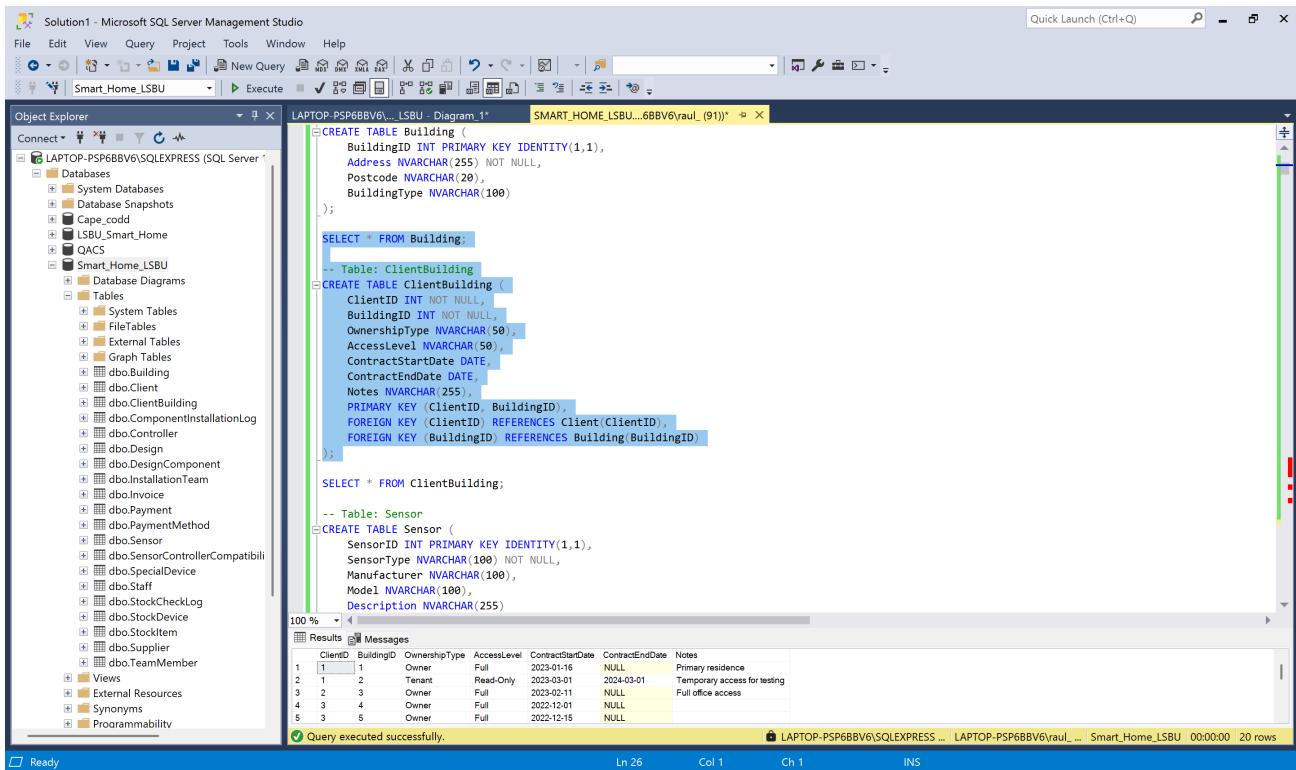
```

INSERT INTO Building (Address, Postcode, BuildingType)
VALUES
('221B Baker Street', 'NW1 6XE', 'Townhouse'),
('14 Liverpool Street', 'EC2M 7PP', 'Office Block'),
('45 Victoria Road', 'W3 6BL', 'Detached House'),
('3A Stratford Avenue', 'E15 4QS', 'Semi-Detached House'),
('101 Camden High Street', 'NW1 7JN', 'Retail and Apartment'),
('7A Oxford Street', 'W1D 2LT', 'Retail Unit'),
('20 Clapham Road', 'SW9 0JG', 'Terraced House'),
('88 Kensington Road', 'W8 5NX', 'Luxury Apartment'),
('12 Shoreditch High Street', 'E1 6PG', 'Loft Conversion'),
('55 Brixton Hill', 'SW2 1HT', 'Apartment Block'),
('100 Whitechapel Road', 'E1 1JG', 'Flat'),
('62 Notting Hill Gate', 'W11 3HT', 'Terraced House'),
('77 Tottenham Court Road', 'W1T 4TJ', 'Office Floor'),
('4 Hackney Road', 'E2 7NX', 'Studio Flat'),
('5 Croydon High Street', 'CR0 1QQ', 'Commercial Unit'),
('39 King's Cross Road', 'WC1X 9LP', 'Shared Housing'),
('15 Bermondsey Street', 'SE1 2BT', 'New Build Apartment'),
('33 Peckham Rye', 'SE15 4JR', 'Maisondette'),
('21 Westbourne Grove', 'W2 4UA', 'Converted Townhouse'),
('6 Elephant Road', 'SE17 1LB', 'Council Flat');

```

The 'Results' tab shows the output of the INSERT statement. It lists 15 rows of building addresses, postcodes, and building types. The last row is a SELECT \* statement.

Figure 6. Insert Building Values



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home LSBU Smart\_Home LSBU...6BBV6\raul\_ (91)\* Execute

Object Explorer

LAPTOP-PSP6BBV6\LSBU - Diagram.1\* SMART\_HOME LSBU...6BBV6\raul\_ (91)\*

```
--CREATE TABLE Building (
    BuildingID INT PRIMARY KEY IDENTITY(1,1),
    Address NVARCHAR(255) NOT NULL,
    Postcode NVARCHAR(20),
    BuildingType NVARCHAR(100)
);

SELECT * FROM Building;

-- Table: ClientBuilding
CREATE TABLE ClientBuilding (
    ClientID INT NOT NULL,
    BuildingID INT NOT NULL,
    OwnershipType NVARCHAR(50),
    AccessLevel NVARCHAR(50),
    ContractStartDate DATE,
    ContractEndDate DATE,
    Notes NVARCHAR(255),
    PRIMARY KEY (ClientID, BuildingID),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID),
    FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID)
);

SELECT * FROM ClientBuilding;

-- Table: Sensor
CREATE TABLE Sensor (
    SensorID INT PRIMARY KEY IDENTITY(1,1),
    SensorType NVARCHAR(100) NOT NULL,
    Manufacturer NVARCHAR(100),
    Model NVARCHAR(100),
    Description NVARCHAR(255)
);
```

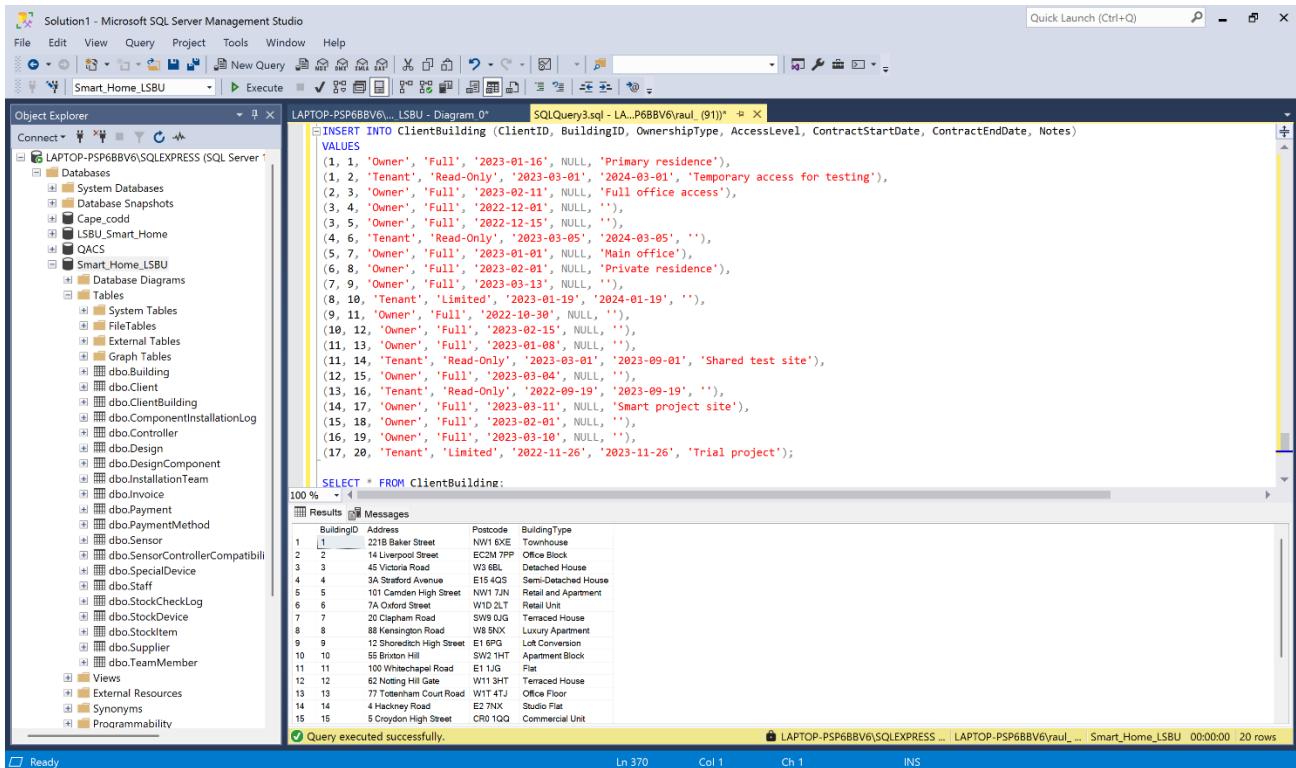
Results

	ClientID	BuildingID	OwnershipType	AccessLevel	ContractStartDate	ContractEndDate	Notes
1	1	1	Owner	Full	2023-01-16	NULL	Primary residence
2	1	2	Tenant	Read-Only	2023-03-01	2024-03-01	Temporary access for testing
3	2	3	Owner	Full	2023-02-11	NULL	Full office access
4	3	4	Owner	Full	2022-12-01	NULL	
5	3	5	Owner	Full	2022-12-15	NULL	

Messages

Query executed successfully.

Figure 7. Create ClientBuilding



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home LSBU Smart\_Home LSBU...6BBV6\raul\_ (91)\* Execute

Object Explorer

LAPTOP-PSP6BBV6\LSBU - Diagram.0\* SQLQuery3.sql - LA...P6BBV6\raul\_ (91)\*

```
INSERT INTO ClientBuilding (ClientID, BuildingID, OwnershipType, AccessLevel, ContractStartDate, ContractEndDate, Notes)
VALUES
(1, 1, 'Owner', 'Full', '2023-01-16', NULL, 'Primary residence'),
(1, 2, 'Tenant', 'Read-Only', '2023-03-01', '2024-03-01', 'Temporary access for testing'),
(2, 3, 'Owner', 'Full', '2023-02-11', NULL, 'Full office access'),
(3, 4, 'Owner', 'Full', '2022-12-01', NULL, ''),
(3, 5, 'Owner', 'Full', '2022-12-15', NULL, ''),
(4, 6, 'Tenant', 'Read-Only', '2023-03-05', '2024-03-05', ''),
(5, 7, 'Owner', 'Full', '2023-01-01', NULL, 'Main office'),
(6, 8, 'Owner', 'Full', '2023-02-01', NULL, 'Private residence'),
(7, 9, 'Owner', 'Full', '2023-03-13', NULL, ''),
(8, 10, 'Tenant', 'Limited', '2023-01-19', '2024-01-19', ''),
(9, 11, 'Owner', 'Full', '2022-10-30', NULL, ''),
(10, 12, 'Owner', 'Full', '2023-02-15', NULL, ''),
(11, 13, 'Owner', 'Full', '2023-01-08', NULL, ''),
(11, 14, 'Tenant', 'Read-Only', '2023-03-01', '2023-09-01', 'Shared test site'),
(12, 15, 'Owner', 'Full', '2023-03-04', NULL, ''),
(13, 16, 'Tenant', 'Read-Only', '2022-09-19', '2023-09-19', ''),
(14, 17, 'Owner', 'Full', '2023-03-11', NULL, 'Smart project site'),
(15, 18, 'Owner', 'Full', '2023-02-01', NULL, ''),
(16, 19, 'Owner', 'Full', '2023-03-10', NULL, ''),
(17, 20, 'Tenant', 'Limited', '2022-11-26', '2023-11-26', 'Trial project');
```

Results

	BuildingID	Address	Postcode	BuildingType
1	1	231B Baker Street	NW1 6XE	Townhouse
2	2	14 Liverpool Street	EC2M 7PP	Office Block
3	3	45 Victoria Road	W3 6BL	Detached House
4	4	3A Stratford Avenue	E15 4QS	Semi-Detached House
5	5	101 Camden High Street	NW1 7JN	Retail and Apartment
6	6	7A Oxford Street	W1D 2LT	Retail Unit
7	7	20 Clapham Road	SW9 0JA	Terraced House
8	8	88 Gloucester Road	SW1 4BY	Luxury Apartment
9	9	12 Shoreham High Street	E1 6PD	Flat Conversion
10	10	55 Brinton Hill	SW2 1HT	Apartment Block
11	11	100 Whitechapel Road	E1 1JG	Flat
12	12	62 Notting Hill Gate	W11 3HT	Terraced House
13	13	77 Tottenham Court Road	W1T 4TJ	Office Floor
14	14	4 Hackney Road	E2 7NK	Studio Flat
15	15	5 Croydon High Street	CR0 1QQ	Commercial Unit

Messages

Query executed successfully.

Figure 8. Insert ClientBuilding Values

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. A query window titled 'LAPTOP-PSP6BBV6...\_LSBU - Diagram.1\*' is open, displaying the creation of a 'Sensor' table. The table has columns: SensorID (INT, primary key identity(1,1)), SensorType (NVARCHAR(100) NOT NULL), Manufacturer (NVARCHAR(100)), Model (NVARCHAR(100)), and Description (NVARCHAR(255)). Below the table definition, there are additional CREATE TABLE statements for 'Controller' and 'SpecialDevice' tables. The results pane shows the inserted sensor data:

SensorID	SensorType	Manufacturer	Model	Description
1	Temperature	Honeywell	THX9421R	High-precision indoor temperature sensor
2	Humidity	Bosch	BME280	Compact humidity and pressure sensor
3	Smoke Detector	Nest	Protect-2ndGen	Smart smoke and CO detector with alerts
4	Motion	Philips Hue	MotionSensorV2	Indoor motion sensor with light level detection
5	DoorWindow Contact	Aqara	MCCGQ11LM	Magnetic contact sensor for doors/windows

Query executed successfully.

Figure 9. Create Sensor

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. A query window titled 'LAPTOP-PSP6BBV6...\_LSBU - Diagram.0\*' is open, displaying an INSERT INTO statement for the 'Sensor' table. The statement inserts 15 rows of sensor data with details like SensorID, SensorType, Manufacturer, Model, and Description. The results pane shows the inserted sensor data:

SensorID	SensorType	Manufacturer	Model	Description
1	Temperature	Honeywell	THX9421R	High-precision indoor temperature sensor
2	Humidity	Bosch	BME280	Compact humidity and pressure sensor
3	Smoke Detector	Nest	Protect-2ndGen	Smart smoke and CO detector with alerts
4	Motion	Philips Hue	MotionSensorV2	Indoor motion sensor with light level detection
5	DoorWindow Contact	Aqara	MCCGQ11LM	Magnetic contact sensor for doors/windows
6	Leak Detection	Fibaro	FGFS-101	Water leak sensor for under sinks and appliances
7	Air Quality	Netatmo	HealthyHomeCoach	Mонitors CO <sub>2</sub> , humidity, noise and temperature
8	Light Sensor	Aeotec	ZWA005	Light level detection for automation
9	Vibration Sensor	Samsung SmartThings	STS-HM-350	Detects vibration or tampering
10	CO2 Sensor	Libellum	PIR2HAWF2-5-ECO	Measures CO <sub>2</sub> for air quality monitoring
11	Cooker Sensor		PIR2HAWF2-5-ECO	Measures CO <sub>2</sub> for air quality monitoring
12	Glass Break Sensor	Honeywell	FG162S	Detects glass breakage with audio analysis
13	UV Sensor	Adafruit	VEML6075	Ultraviolet light detection for indoor exposure
14	Sound Sensor	Grove	SEN-12462	Monitors ambient sound levels
15	Flood Sensor	D-Link	DCH-S162	Wi-Fi water leak and freeze detector

Query executed successfully.

Figure 10. Insert Sensor Values

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The central pane displays a T-SQL script for creating a 'Controller' table:

```

Model NVARCHAR(100),
Description NVARCHAR(255)
);

SELECT * FROM Sensor;

-- Table: Controller
CREATE TABLE Controller (
    ControllerID INT PRIMARY KEY IDENTITY(1,1),
    ControllerType NVARCHAR(100) NOT NULL,
    Protocol NVARCHAR(50),
    Manufacturer NVARCHAR(100),
    Model NVARCHAR(100)
);

SELECT * FROM Controller;

-- Table: SpecialDevice
CREATE TABLE SpecialDevice (
    SpecialDeviceID INT PRIMARY KEY IDENTITY(1,1),
    DeviceType NVARCHAR(100) NOT NULL,
    Manufacturer NVARCHAR(100),
    Model NVARCHAR(100),
    Description NVARCHAR(255)
);

SELECT * FROM SpecialDevice;

-- Table: Design
CREATE TABLE Design (

```

The results pane shows the execution status: "Query executed successfully." Below the results, a table is displayed with 5 rows of data:

	ControllerID	ControllerType	Protocol	Manufacturer	Model
1	1	Smart Hub	Zigbee	Samsung SmartThings	STH-ETH-250
2	2	Bridge	Zigbee	Philips Hue	Hue Bridge v2
3	3	Home Gateway	Z-Wave	Aeotec	Z-Stick Gen5
4	4	IoT Edge Gateway	Wi-Fi	Cisco	IR1101
5	5	Smart Hub	Wi-Fi	Amazon	Echo 4th Gen

Figure 11.Create Controller

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The central pane displays an 'INSERT INTO' statement for the 'Controller' table:

```

VALUES
('Smart Hub', 'Zigbee', 'Samsung SmartThings', 'STH-ETH-250'),
('Bridge', 'Zigbee', 'Philips Hue', 'Hue Bridge v2'),
('Home Gateway', 'Z-Wave', 'Aeotec', 'Z-Stick Gen5'),
('IoT Edge Gateway', 'Wi-Fi', 'Cisco', 'IR1101'),
('Smart Hub', 'Wi-Fi', 'Amazon', 'Echo 4th Gen'),
('Controller', 'Z-Wave', 'Fibaro', 'Home Center 3'),
('Hub', 'Thread', 'Google Nest', 'Nest Hub Max'),
('Central Unit', 'LoRa', 'Lilium', 'Meshlium Xtreme'),
('Gateway', 'Bluetooth', 'Xiaomi', 'Aqara Hub M2'),
('Edge Controller', 'Ethernet', 'Bosch', 'IoT Gateway'),
('Smart Gateway', 'Zigbee', 'Tuya', 'Zigbee 3.0 Hub'),
('Router + Hub', 'Zigbee/Z-Wave', 'Hubitat', 'Elevation C-8'),
('IoT Hub', 'Wi-Fi', 'TP-Link', 'Kasa Smart Hub KH100'),
('Security Panel', 'Z-Wave', 'Ring', 'Alarm Base Station Gen2'),
('Bridge', 'Zigbee', 'IKEA', 'TRADFRI Gateway'),
('IoT Gateway', 'LTE', 'Teltonika', 'TRB140'),
('Mini Controller', 'Wi-Fi', 'Sonoff', 'NSPanel'),
('Mesh Controller', 'Z-Wave', 'Ezlo', 'Plug Hub'),
('Smart Panel', 'Zigbee', 'Moes', 'Zigbee Scene Controller'),
('IoT Edge Box', 'Modbus TCP', 'Advantech', 'UNO-420');

```

The results pane shows the execution status: "Query executed successfully." Below the results, a table is displayed with 15 rows of data:

	ControllerID	ControllerType	Protocol	Manufacturer	Model
1	1	Smart Hub	Zigbee	Samsung SmartThings	STH-ETH-250
2	2	Bridge	Zigbee	Philips Hue	Hue Bridge v2
3	3	Home Gateway	Z-Wave	Aeotec	Z-Stick Gen5
4	4	IoT Edge Gateway	Wi-Fi	Cisco	IR1101
5	5	Smart Hub	Wi-Fi	Amazon	Echo 4th Gen
6	6	Controller	Z-Wave	Fibaro	Home Center 3
7	7	Hub	Thread	Google Nest	Nest Hub Max
8	8	Central Unit	LoRa	Lilium	Meshlium Xtreme
9	9	Gateway	Bluetooth	Xiaomi	Aqara Hub M2
10	10	Edge Controller	Ethernet	Bosch	IoT Gateway
11	11	Smart Gateway	Zigbee	Tuya	Zigbee 3.0 Hub
12	12	Router + Hub	Zigbee/Z-Wave	Hubitat	Elevation C-8
13	13	IoT Hub	Wi-Fi	TP-Link	Kasa Smart Hub KH100
14	14	Security Panel	Z-Wave	Ring	Alarm Base Station Gen2
15	15	Bridge	Zigbee	IKEA	TRADFRI Gateway

Figure 12.Insert Controller Values

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The main pane displays the creation script for the 'SpecialDevice' table:

```

Model NVARCHAR(100),
Description NVARCHAR(255)
);

SELECT * FROM Sensor;

-- Table: Controller
CREATE TABLE Controller (
    ControllerID INT PRIMARY KEY IDENTITY(1,1),
    ControllerType NVARCHAR(100) NOT NULL,
    Protocol NVARCHAR(50),
    Manufacturer NVARCHAR(100),
    Model NVARCHAR(100)
);

SELECT * FROM Controller;

-- Table: SpecialDevice
CREATE TABLE SpecialDevice (
    SpecialDeviceID INT PRIMARY KEY IDENTITY(1,1),
    DeviceType NVARCHAR(100) NOT NULL,
    Manufacturer NVARCHAR(100),
    Model NVARCHAR(100),
    Description NVARCHAR(255)
);

SELECT * FROM SpecialDevice;

-- Table: Design
CREATE TABLE Design (

```

The results pane shows the execution status: "Query executed successfully." Below the results, a table displays the inserted data:

SpecialDeviceID	DeviceType	Manufacturer	Model	Description
1	HD CCTV Camera	Ario	Pro 4	1080p wire-free security camera with night vision
2	Smart Door Lock	August	Wi-Fi Smart Lock	Keyless entry with mobile control
3	Smart Thermostat	Nest	Learning Thermostat Gen 3	Leans schedule and controls temperature
4	Smart Plug	TP-Link	HS110	Wi-Fi plug with energy monitoring
5	Smart Light Switch	Lutron	Caseta Wireless	Wireless smart light dimmer

Figure 13. Create SpecialDevice

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The main pane displays the insert statement for the 'SpecialDevice' table:

```

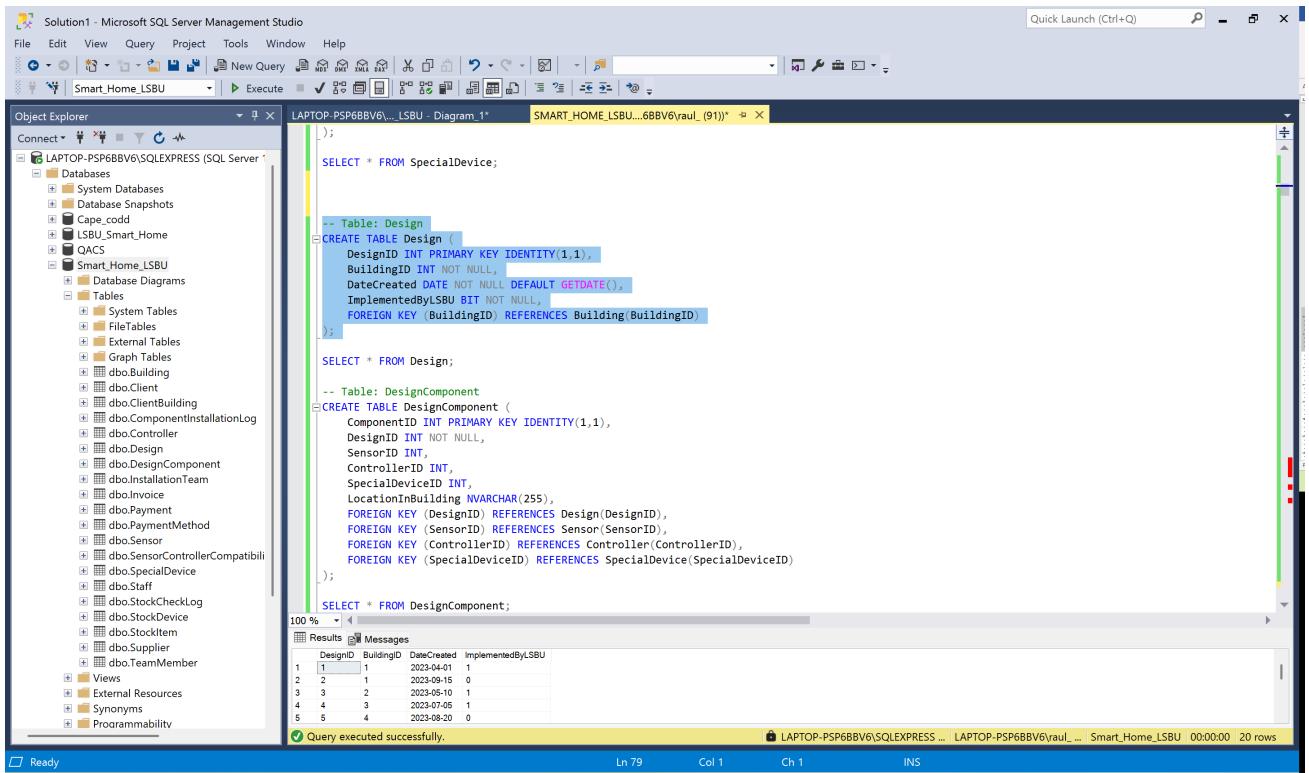
INSERT INTO SpecialDevice (DeviceType, Manufacturer, Model, Description)
VALUES
('HD CCTV Camera', 'Ario', 'Pro 4', '1080p wire-free security camera with night vision'),
('Smart Door Lock', 'August', 'Wi-Fi Smart Lock', 'Keyless entry with mobile control'),
('Smart Thermostat', 'Nest', 'Learning Thermostat Gen 3', 'Leans schedule and controls temperature'),
('Smart Plug', 'TP-Link', 'HS110', 'Wi-Fi plug with energy monitoring'),
('Smart Light Switch', 'Lutron', 'Caseta Wireless', 'Wireless smart light dimmer'),
('Smart Doorbell', 'Ring', 'Video Doorbell Pro 2', 'HD video with motion detection and audio'),
('Smart Alarm Siren', 'Aeotec', 'Z-Wave Siren 6', 'Loud alarm with visual alerts'),
('Automation Panel', 'Fibaro', 'Wall Controller', 'Scene activation panel for automation'),
('Smart Curtain Controller', 'Aqara', 'Roller Shade Driver E1', 'Controls motorized blinds'),
('Garage Door Opener', 'Meross', 'MSG100', 'Wi-Fi smart garage controller'),
('Ultrasonic Motion Detector', 'Milesight', 'EM300-UDL', 'Ultrasonic level & motion sensor'),
('IR Blaster', 'Broadlink', 'RM4 Pro', 'Controls IR and RF home devices'),
('Smart Leak Shutoff Valve', 'Guardian', 'Leak Prevention System', 'Detects leaks and shuts off water'),
('Smart Fan Controller', 'Sensibo', 'Sky', 'Wi-Fi AC and fan controller'),
('Smart Energy Monitor', 'Shelly', 'SEM', 'Three-phase electricity monitoring device'),
('Smart Window Opener', 'Opalux', 'OPWX-01', 'Motorized window opener with remote'),
('Smart Intercom System', '2M', 'IP Verso', 'Smart video/audio intercom for buildings'),
('IR Temperature Scanner', 'Seek', 'CompactPRO', 'Thermal imaging for entry points'),
('Smart Pet Feeder', 'PetSafe', 'Smart Feeder 2nd Gen', 'Automatic feeder with Wi-Fi app control'),
('Smart Irrigation Controller', 'Rachio', 'Rachio 3', 'Wi-Fi sprinkler controller with weather sync');

```

The results pane shows the execution status: "Query executed successfully." Below the results, a table displays the inserted data:

SpecialDeviceID	DeviceType	Manufacturer	Model	Description
1	HD CCTV Camera	Ario	Pro 4	1080p wire-free security camera with night vision
2	Smart Door Lock	August	Wi-Fi Smart Lock	Keyless entry with mobile control
3	Smart Thermostat	Nest	Learning Thermostat Gen 3	Leans schedule and controls temperature
4	Smart Plug	TP-Link	HS110	Wi-Fi plug with energy monitoring
5	Smart Light Switch	Lutron	Caseta Wireless	Wireless smart light dimmer
6	Smart Doorbell	Ring	Video Doorbell Pro 2	HD video with motion detection and audio
7	Smart Alarm Siren	Aeotec	Z-Wave Siren 6	Loud alarm with visual alerts
8	Automation Panel	Fibaro	Wall Controller	Scene activation panel for automation
9	Smart Curtain Controller	Aqara	Roller Shade Driver E1	Controls motorized blinds
10	Garage Door Opener	Meross	MSG100	Wi-Fi smart garage controller
11	Ultrasonic Motion Detector	Milesight	EM300-UDL	Ultrasonic level & motion sensor
12	IR Blaster	Broadlink	RM4 Pro	Controls IR and RF home devices
13	Smart Leak Shutoff Valve	Guardian	Leak Prevention System	Detects leaks and shuts off water
14	Smart Fan Controller	Sensibo	Sky	Wi-Fi AC and fan controller
15	Smart Energy Monitor	Shelly	SEM	Three-phase electricity monitoring device

Figure 14. Insert SpecialDevice Values



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home LSBU'. The central pane displays a T-SQL script for creating the 'Design' table:

```

CREATE TABLE Design (
    DesignID INT PRIMARY KEY IDENTITY(1,1),
    BuildingID INT NOT NULL,
    DateCreated DATE NOT NULL DEFAULT GETDATE(),
    ImplementedByLSBU BIT NOT NULL,
    FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID)
);

SELECT * FROM Design;

```

Below this, another section of code creates the 'DesignComponent' table:

```

CREATE TABLE DesignComponent (
    ComponentID INT PRIMARY KEY IDENTITY(1,1),
    DesignID INT NOT NULL,
    SensorID INT,
    ControllerID INT,
    SpecialDeviceID INT,
    LocationInBuilding NVARCHAR(255),
    FOREIGN KEY (DesignID) REFERENCES Design(DesignID),
    FOREIGN KEY (SensorID) REFERENCES Sensor(SensorID),
    FOREIGN KEY (ControllerID) REFERENCES Controller(ControllerID),
    FOREIGN KEY (SpecialDeviceID) REFERENCES SpecialDevice(SpecialDeviceID)
);

SELECT * FROM DesignComponent;

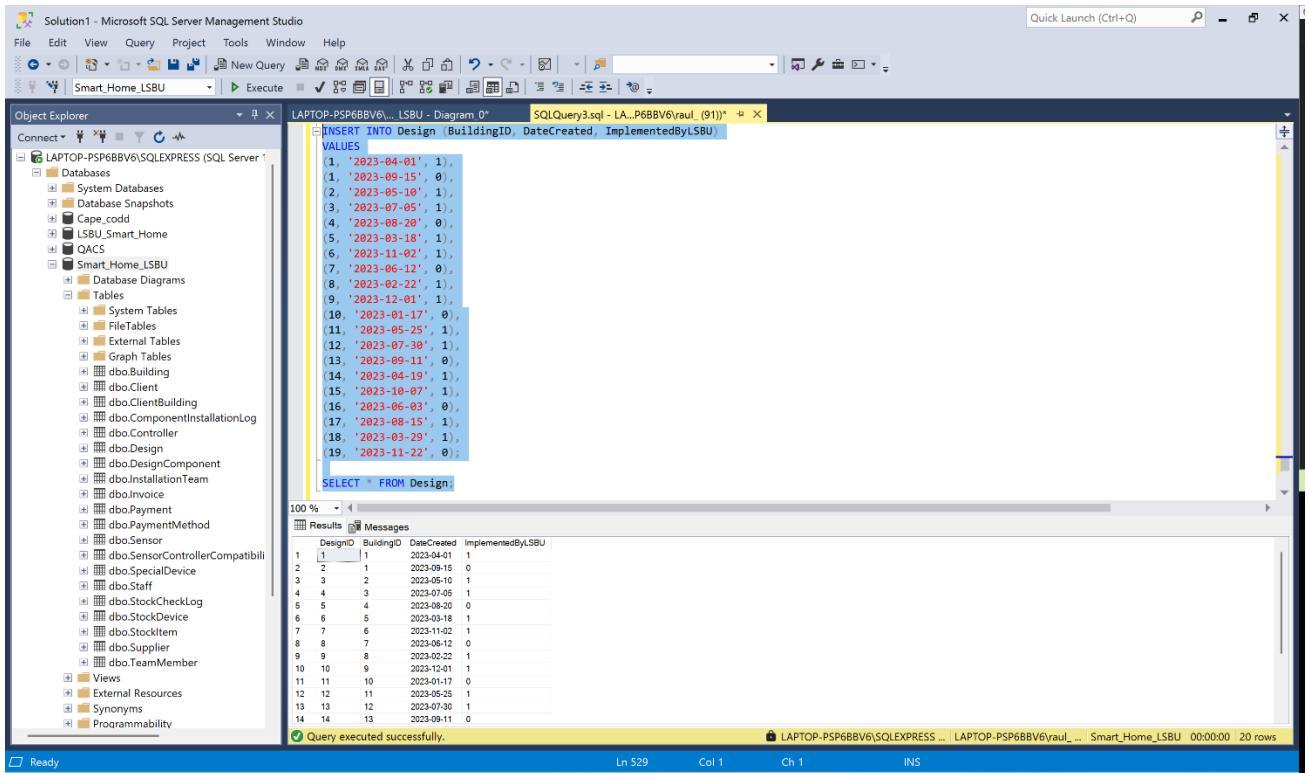
```

The results pane shows the output of the last query:

	DesignID	BuildingID	DateCreated	ImplementedByLSBU
1	1	1	2023-04-01	1
2	2	1	2023-09-15	0
3	3	2	2023-05-10	1
4	4	3	2023-07-05	1
5	5	4	2023-08-20	0

A message at the bottom indicates: "Query executed successfully."

Figure 15. Create Design



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home LSBU'. The central pane displays an INSERT INTO statement for the 'Design' table:

```

INSERT INTO Design (BuildingID, DateCreated, ImplementedByLSBU)
VALUES
    ('2023-04-01', 1),
    ('2023-09-15', 0),
    ('2023-05-10', 1),
    ('2023-07-05', 1),
    ('2023-08-20', 0),
    ('2023-03-18', 1),
    ('2023-11-02', 1),
    ('2023-06-12', 0),
    ('2023-02-22', 1),
    ('2023-12-01', 1),
    ('2023-01-17', 0),
    ('2023-05-25', 1),
    ('2023-07-30', 1),
    ('2023-09-11', 0),
    ('2023-04-19', 1),
    ('2023-10-07', 1),
    ('2023-06-03', 0),
    ('2023-08-15', 1),
    ('2023-03-29', 1),
    ('2023-11-22', 0);

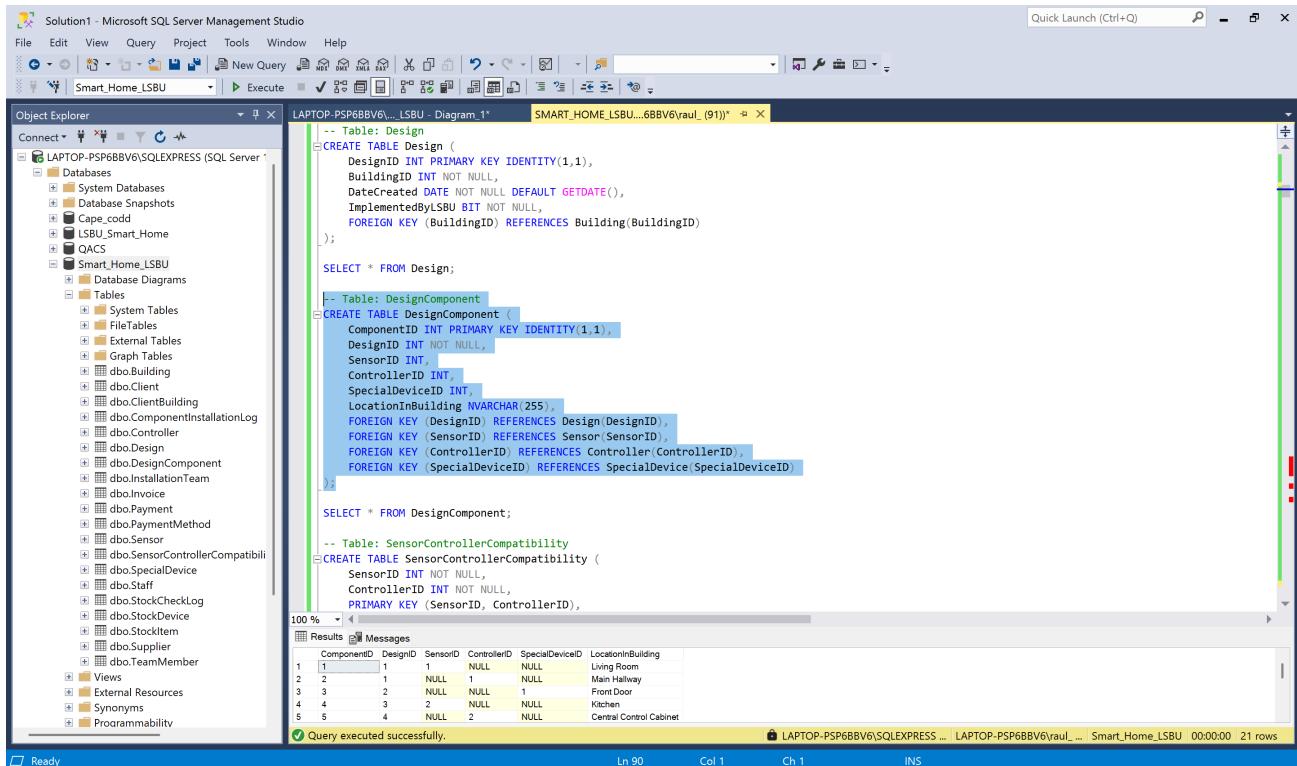
```

The results pane shows the output of the last query:

	DesignID	BuildingID	DateCreated	ImplementedByLSBU
1	1	1	2023-04-01	1
2	2	1	2023-09-15	0
3	3	2	2023-05-10	1
4	4	3	2023-07-05	1
5	5	4	2023-03-18	0
6	6	5	2023-11-02	1
7	7	6	2023-06-12	0
8	8	7	2023-02-22	1
9	9	8	2023-12-01	1
10	10	9	2023-05-25	0
11	11	10	2023-07-30	1
12	12	11	2023-09-11	0
13	13	12	2023-04-19	1
14	14	13	2023-10-07	1

A message at the bottom indicates: "Query executed successfully."

Figure 16. Insert Design Values



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home\_LSBU Execute

Object Explorer

LAPTOP-PSP6BBV6\LSBU - Diagram: 1\* SMART\_HOME\_LSBU...68BV6\raul\_(91)\*

```
-- Table: Design
CREATE TABLE Design (
    DesignID INT PRIMARY KEY IDENTITY(1,1),
    BuildingID INT NOT NULL,
    DateCreated DATE NOT NULL DEFAULT GETDATE(),
    ImplementedByLSBU BIT NOT NULL,
    FOREIGN KEY (BuildingID) REFERENCES Building(BuildingID)
);

SELECT * FROM Design;

-- Table: DesignComponent
CREATE TABLE DesignComponent (
    ComponentID INT PRIMARY KEY IDENTITY(1,1),
    DesignID INT NOT NULL,
    SensorID INT,
    ControllerID INT,
    SpecialDeviceID INT,
    LocationInBuilding NVARCHAR(255),
    FOREIGN KEY (DesignID) REFERENCES Design(DesignID),
    FOREIGN KEY (SensorID) REFERENCES Sensor(SensorID),
    FOREIGN KEY (ControllerID) REFERENCES Controller(ControllerID),
    FOREIGN KEY (SpecialDeviceID) REFERENCES SpecialDevice(SpecialDeviceID)
);

SELECT * FROM DesignComponent;
```

-- Table: SensorControllerCompatibility

```
CREATE TABLE SensorControllerCompatibility (
    SensorID INT NOT NULL,
    ControllerID INT NOT NULL,
    PRIMARY KEY (SensorID, ControllerID),
    FOREIGN KEY (SensorID) REFERENCES Sensor(SensorID),
    FOREIGN KEY (ControllerID) REFERENCES Controller(ControllerID)
);
```

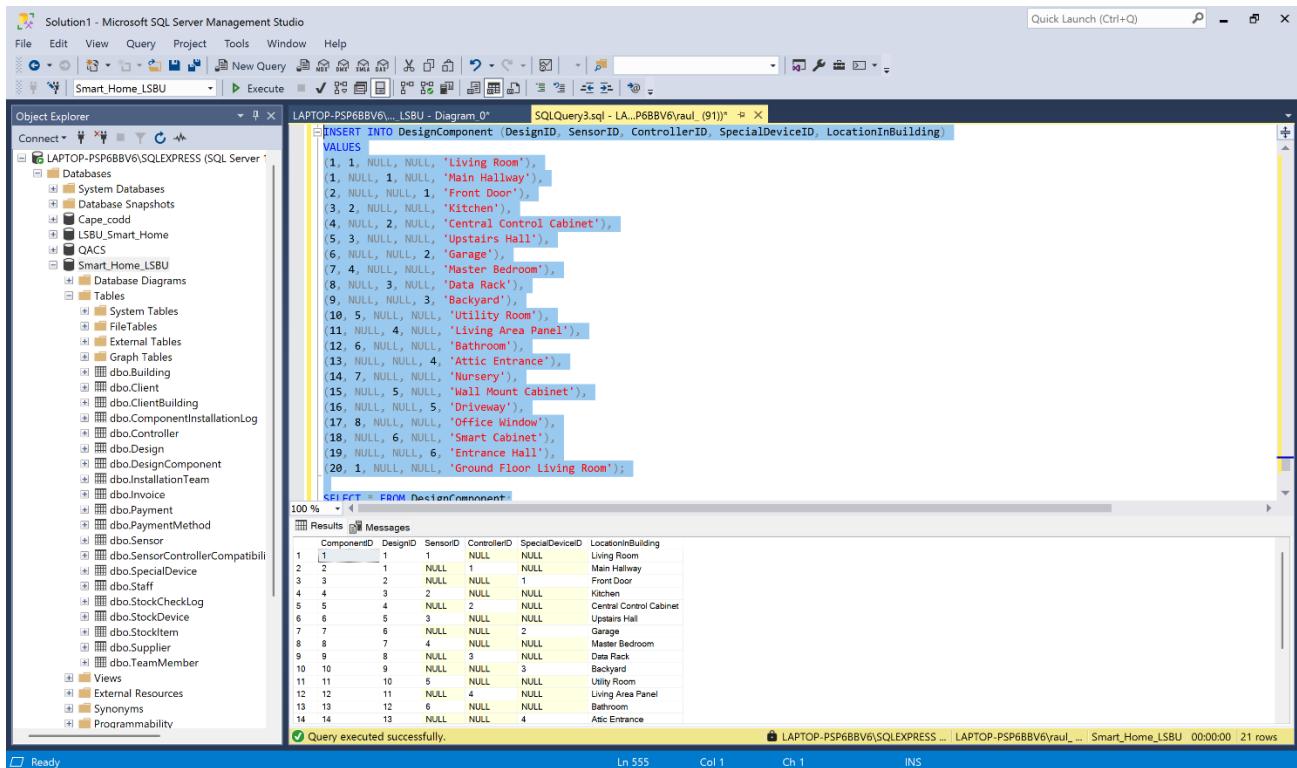
Result Messages

ComponentID	DesignID	SensorID	ControllerID	SpecialDeviceID	LocationInBuilding
1	1	1	NULL	NULL	Living Room
2	2	1	NULL	1	Main Hallway
3	3	2	NULL	1	Front Door
4	4	3	2	NULL	Kitchen
5	5	4	NULL	2	Central Control Cabinet

Query executed successfully.

LAPTOP-PSP6BBV6\SQLEXPRESS ... LAPTOP-PSP6BBV6\raul\_ ... Smart\_Home\_LSBU 00:00:00 21 rows

Figure 17. Create DesignComponent



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home\_LSBU Execute

Object Explorer

LAPTOP-PSP6BBV6\LSBU - Diagram: 0\* SQLQuery3.sql - LA...PSP6BBV6\raul\_(91)\*

```
INSERT INTO DesignComponent (DesignID, SensorID, ControllerID, SpecialDeviceID, LocationInBuilding)
VALUES
(1, 1, NULL, 'Living Room'),
(1, NULL, 1, NULL, 'Main Hallway'),
(2, NULL, NULL, 1, 'Front Door'),
(3, 2, NULL, NULL, 'Kitchen'),
(4, NULL, 2, NULL, 'Central Control Cabinet'),
(5, 3, NULL, NULL, 'Upstairs Hall'),
(6, NULL, NULL, 2, 'Garage'),
(7, 4, NULL, NULL, 'Master Bedroom'),
(8, NULL, 3, NULL, 'Data Rack'),
(9, NULL, NULL, 3, 'Backyard'),
(10, 5, NULL, NULL, 'Utility Room'),
(11, NULL, 4, NULL, 'Living Area Panel'),
(12, 6, NULL, NULL, 'Bathroom'),
(13, NULL, NULL, 4, 'Attic Entrance'),
(14, 7, NULL, NULL, 'Nursery'),
(15, NULL, 5, NULL, 'Wall Mount Cabinet'),
(16, NULL, NULL, 5, 'Driveway'),
(17, 8, NULL, NULL, 'Office Window'),
(18, NULL, 6, NULL, 'Smart Cabinet'),
(19, NULL, NULL, 6, 'Entrance Hall'),
(20, 1, NULL, NULL, 'Ground Floor Living Room');
```

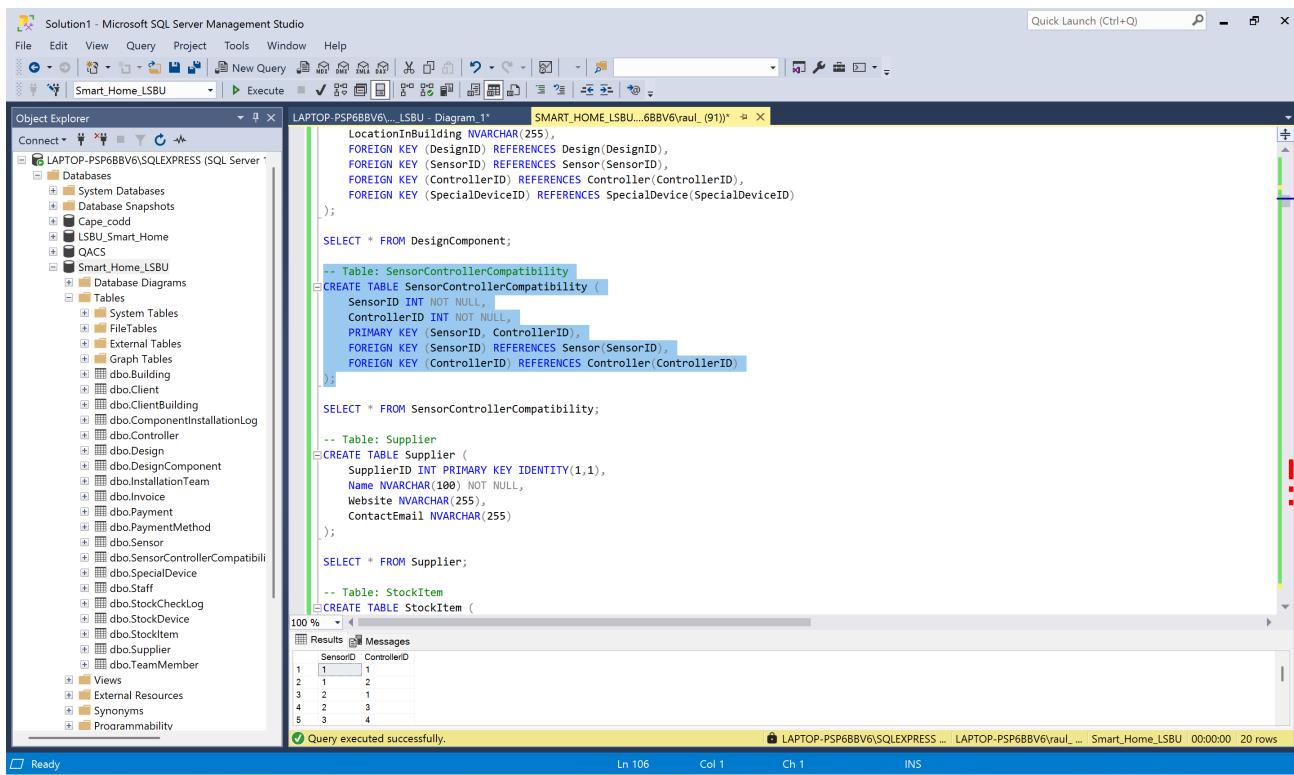
Result Messages

ComponentID	DesignID	SensorID	ControllerID	SpecialDeviceID	LocationInBuilding
1	1	1	NULL	NULL	Living Room
2	2	1	NULL	1	Main Hallway
3	3	2	NULL	1	Front Door
4	4	3	2	NULL	Kitchen
5	5	4	NULL	2	Central Control Cabinet
6	6	5	3	NULL	Upstairs Hall
7	7	6	NULL	2	Garage
8	8	7	NULL	NULL	Master Bedroom
9	9	8	NULL	3	Backyard
10	10	9	NULL	3	Driveway
11	11	10	5	NULL	Utility Room
12	12	11	4	NULL	Living Area Panel
13	13	12	6	NULL	Bathroom
14	14	13	NULL	4	Attic Entrance

Query executed successfully.

LAPTOP-PSP6BBV6\SQLEXPRESS ... LAPTOP-PSP6BBV6\raul\_ ... Smart\_Home\_LSBU 00:00:00 21 rows

Figure 18. Insert DesignComponent Values



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home\_LSBU'. The central pane displays the following T-SQL code:

```

CREATE TABLE SensorControllerCompatibility (
    SensorID INT NOT NULL,
    ControllerID INT NOT NULL,
    PRIMARY KEY (SensorID, ControllerID),
    FOREIGN KEY (SensorID) REFERENCES Sensor(SensorID),
    FOREIGN KEY (ControllerID) REFERENCES Controller(ControllerID)
);

SELECT * FROM SensorControllerCompatibility;

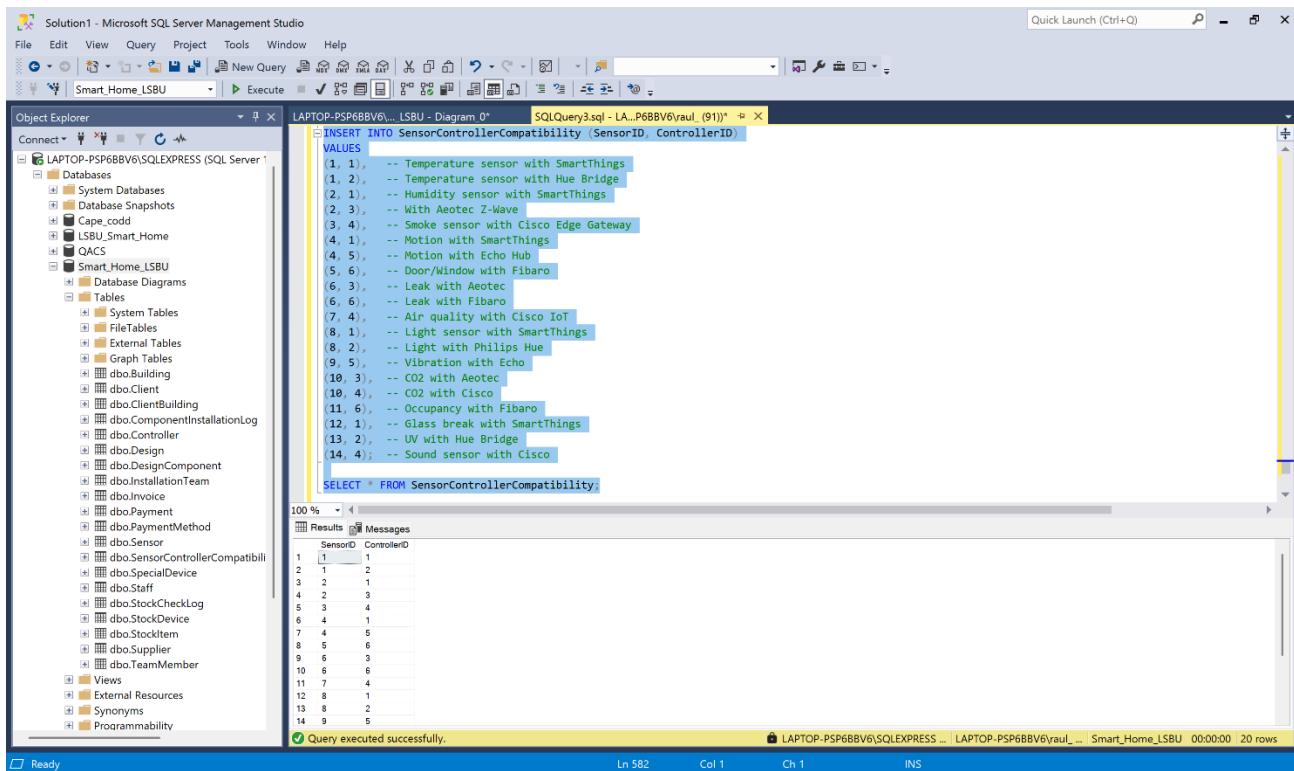
```

The results pane shows the following data:

SensorID	ControllerID
1	1
2	1
3	2
4	2
5	3
6	4

A message at the bottom indicates: "Query executed successfully."

Figure 19. Create SensorControllerCompatibility



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home\_LSBU'. The central pane displays the following T-SQL code:

```

INSERT INTO SensorControllerCompatibility (SensorID, ControllerID)
VALUES
(1, 1), -- Temperature sensor with SmartThings
(1, 2), -- Temperature sensor with Hue Bridge
(2, 1), -- Humidity sensor with SmartThings
(2, 3), -- With Aeotec Z-Wave
(3, 4), -- Smoke sensor with Cisco Edge Gateway
(4, 1), -- Motion with SmartThings
(4, 5), -- Motion with Echo Hub
(5, 6), -- Door/Window with Fibaro
(6, 3), -- Leak with Aeotec
(6, 6), -- Leak with Fibaro
(7, 4), -- Air quality with Cisco IoT
(8, 1), -- Light sensor with SmartThings
(8, 2), -- Light with Philips Hue
(9, 5), -- Vibration with Ech
(10, 3), -- CO2 with Aeotec
(10, 4), -- CO2 with Cisco
(11, 6), -- Occupancy with Fibaro
(12, 1), -- Glass break with SmartThings
(13, 2), -- UV with Hue Bridge
(14, 4); -- Sound sensor with Cisco

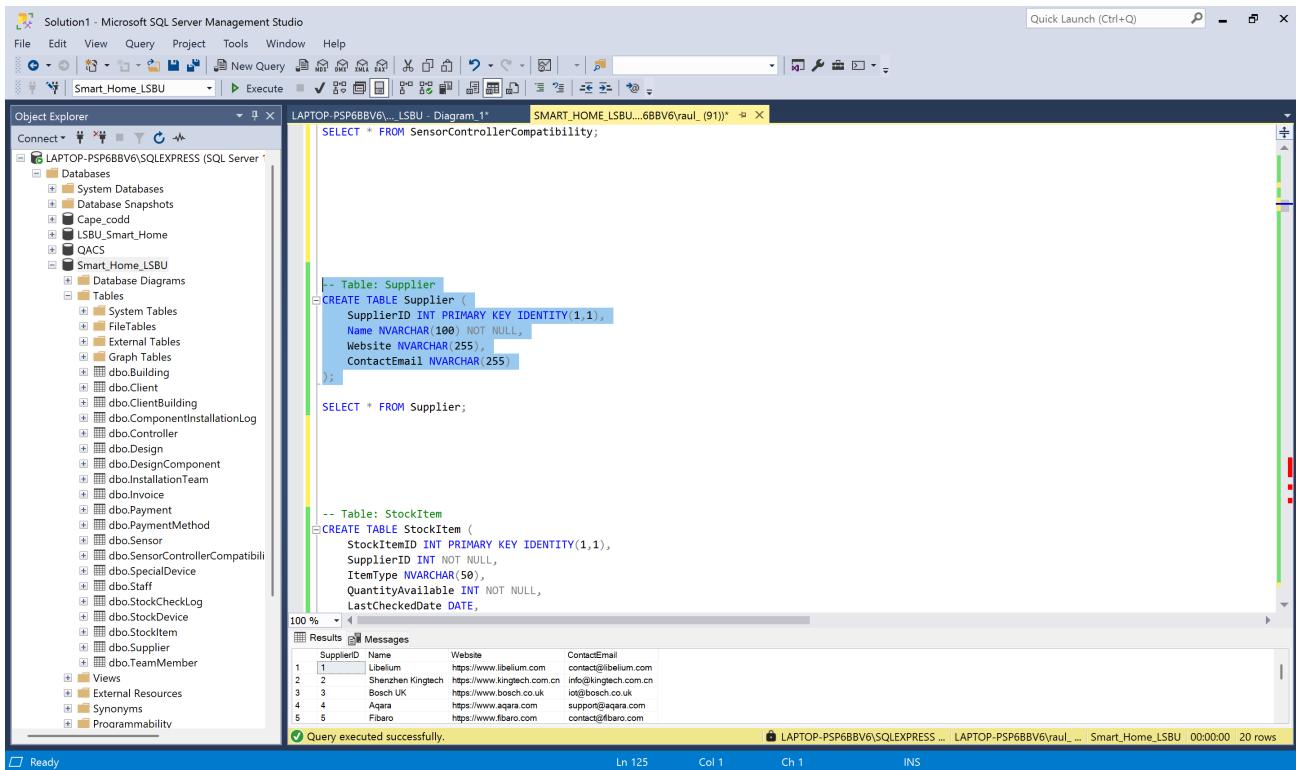
```

The results pane shows the following data:

SensorID	ControllerID
1	1
2	1
3	1
4	2
5	3
6	4
7	4
8	5
9	6
10	6
11	7
12	8
13	8
14	9

A message at the bottom indicates: "Query executed successfully."

Figure 20. Insert SensorControllerCompatibility Values



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home\_LSBU | Execute

Object Explorer

LAPTOP-PSP6BBV6\...\_LSBU - Diagram.1\* SMART\_HOME\_LSBU...6BBV6\raul\_(91)\*

```

SELECT * FROM SensorControllerCompatibility;

-- Table: Supplier
CREATE TABLE Supplier (
    SupplierID INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(100) NOT NULL,
    Website NVARCHAR(255),
    ContactEmail NVARCHAR(255)
);

SELECT * FROM Supplier;

-- Table: StockItem
CREATE TABLE StockItem (
    StockItemID INT PRIMARY KEY IDENTITY(1,1),
    SupplierID INT NOT NULL,
    ItemType NVARCHAR(50),
    QuantityAvailable INT NOT NULL,
    LastCheckedDate DATE
);

```

Results

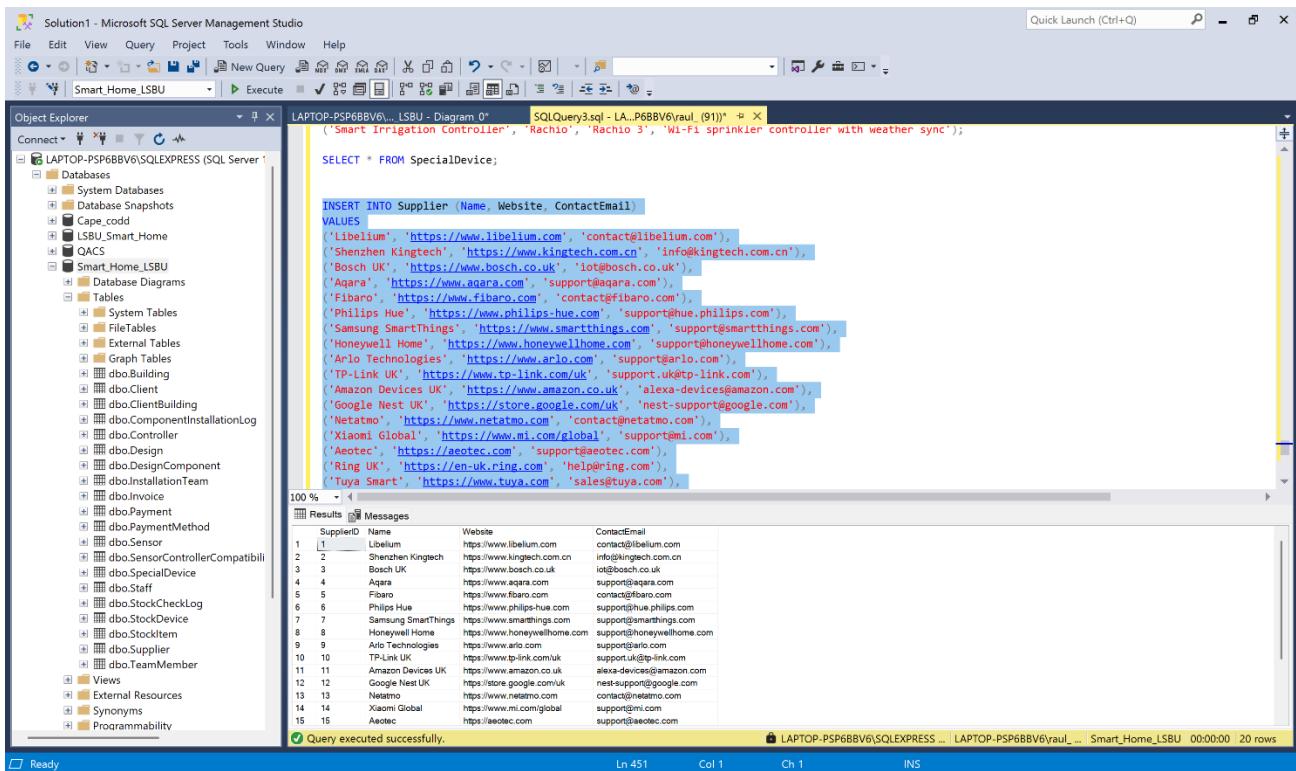
SupplierID	Name	Website	ContactEmail
1	Libelium	https://www.libelium.com	contact@libelium.com
2	Shenzhen Kingtech	https://www.kingtech.com.cn	info@kingtech.com.cn
3	Bosch UK	https://www.bosch.co.uk	io@bosch.co.uk
4	Aqara	https://www.aqara.com	support@aqara.com
5	Fibaro	https://www.fibaro.com	contact@fibaro.com

Messages

Query executed successfully.

Ln 125 Col 1 Ch 1 INS

Figure 21. Create Supplier



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home\_LSBU | Execute

Object Explorer

LAPTOP-PSP6BBV6\...\_LSBU - Diagram 0\* SQLQuery3.sql - LA...PSP6BBV6\raul\_(91)\*

```

('Smart Irrigation Controller', 'Rachio', 'Rachio 3', 'Wi-Fi sprinkler controller with weather sync');

SELECT * FROM SpecialDevice;

```

INSERT INTO Supplier (Name, Website, ContactEmail)

```

VALUES
('Libelium', 'https://www.libelium.com', 'contact@libelium.com'),
('Shenzhen Kingtech', 'https://www.kingtech.com.cn', 'info@kingtech.com.cn'),
('Bosch UK', 'https://www.bosch.co.uk', 'io@bosch.co.uk'),
('Aqara', 'https://www.aqara.com', 'support@aqara.com'),
('Fibaro', 'https://www.fibaro.com', 'contact@fibaro.com'),
('Philips Hue', 'https://www.philips-hue.com', 'support@hue.philips.com'),
('Samsung SmartThings', 'https://www.smarththings.com', 'support@smarththings.com'),
('Honeywell Home', 'https://www.honeywellhome.com', 'support@honeywellhome.com'),
('Arlo Technologies', 'https://www.arlo.com', 'support@arlo.com'),
('TP-Link UK', 'https://www.tp-link.com/uk', 'support.uk@tp-link.com'),
('Amazon Devices UK', 'https://www.amazon.co.uk', 'alexa-devices@amazon.com'),
('Google Nest UK', 'https://store.google.com/uk', 'nest-support@google.com'),
('Netatmo', 'https://www.netatmo.com', 'contact@netatmo.com'),
('Xiaomi Global', 'https://www.mi.com/global', 'support@mi.com'),
('Aeotec', 'https://aeotec.com', 'support@aeotec.com'),
('Ring UK', 'https://en-uk.ring.com', 'help@ring.com'),
('Tuya Smart', 'https://www.tuya.com', 'sales@tuya.com')

```

Results

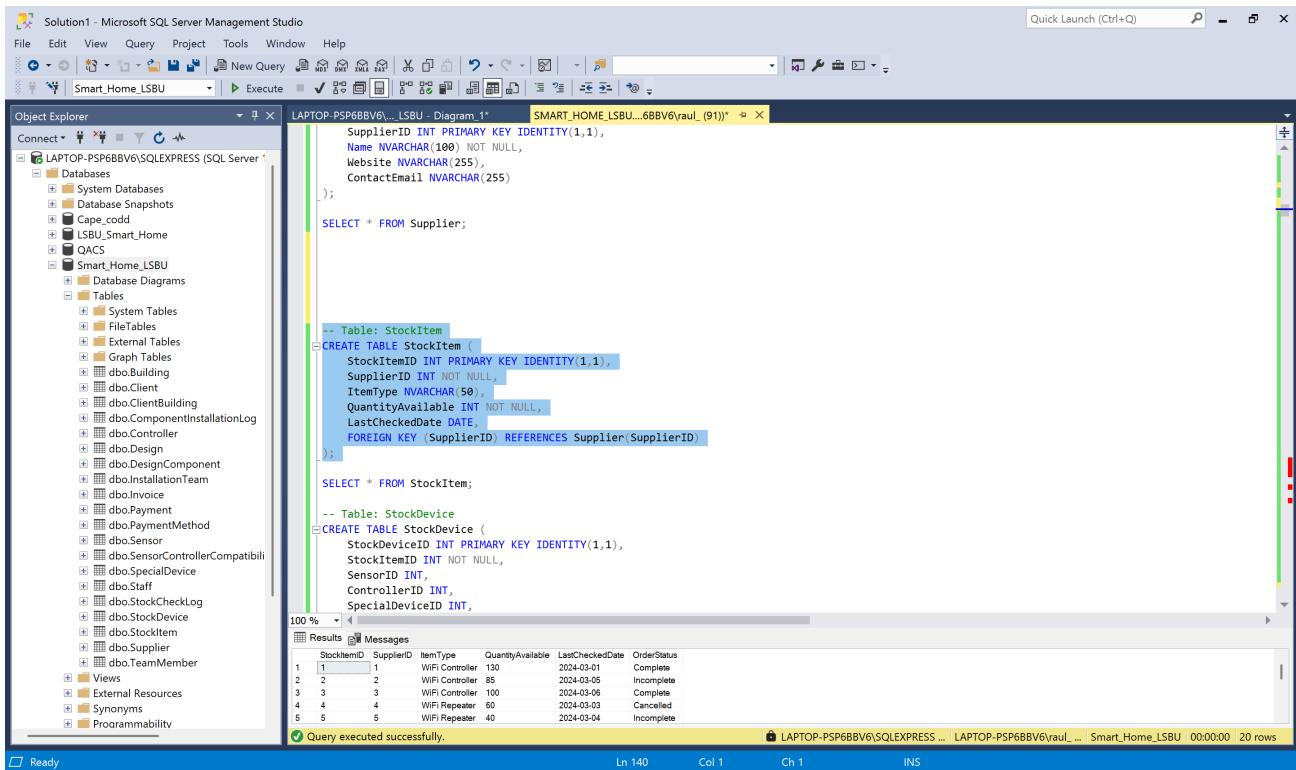
SupplierID	Name	Website	ContactEmail
1	Libelium	https://www.libelium.com	contact@libelium.com
2	Shenzhen Kingtech	https://www.kingtech.com.cn	info@kingtech.com.cn
3	Bosch UK	https://www.bosch.co.uk	io@bosch.co.uk
4	Aqara	https://www.aqara.com	support@aqara.com
5	Fibaro	https://www.fibaro.com	contact@fibaro.com
6	Philips Hue	https://www.philips-hue.com	support@hue.philips.com
7	Samsung SmartThings	https://www.smarththings.com	support@smarththings.com
8	Honeywell Home	https://www.honeywellhome.com	support@honeywellhome.com
9	Arlo Technologies	https://www.arlo.com	support@arlo.com
10	TP-Link UK	https://www.tp-link.com/uk	support.uk@tp-link.com
11	Amazon Devices UK	https://www.amazon.co.uk	alexa-devices@amazon.com
12	Google Nest UK	https://store.google.com/uk	nest-support@google.com
13	Netatmo	https://www.netatmo.com	contact@netatmo.com
14	Xiaomi Global	https://www.mi.com/global	support@mi.com
15	Aeotec	https://aeotec.com	support@aeotec.com

Messages

Query executed successfully.

Ln 451 Col 1 Ch 1 INS

Figure 22. Insert Supplier Values



Solution1 - Microsoft SQL Server Management Studio

LAPTOP-PSP6BBV6\LSBU - Diagram\_1\* SMART\_HOME LSBU...6BBV6\raul\_(91)\*

```

CREATE TABLE StockItem (
    StockItemID INT PRIMARY KEY IDENTITY(1,1),
    SupplierID INT NOT NULL,
    Name NVARCHAR(100),
    Website NVARCHAR(255),
    ContactEmail NVARCHAR(255)
);

SELECT * FROM StockItem;

-- Table: StockDevice
CREATE TABLE StockDevice (
    StockDeviceID INT PRIMARY KEY IDENTITY(1,1),
    StockItemID INT NOT NULL,
    ItemTypeID INT,
    SensorID INT,
    ControllerID INT,
    SpecialDeviceID INT,
    FOREIGN KEY (StockItemID) REFERENCES StockItem(StockItemID)
);

SELECT * FROM StockDevice;

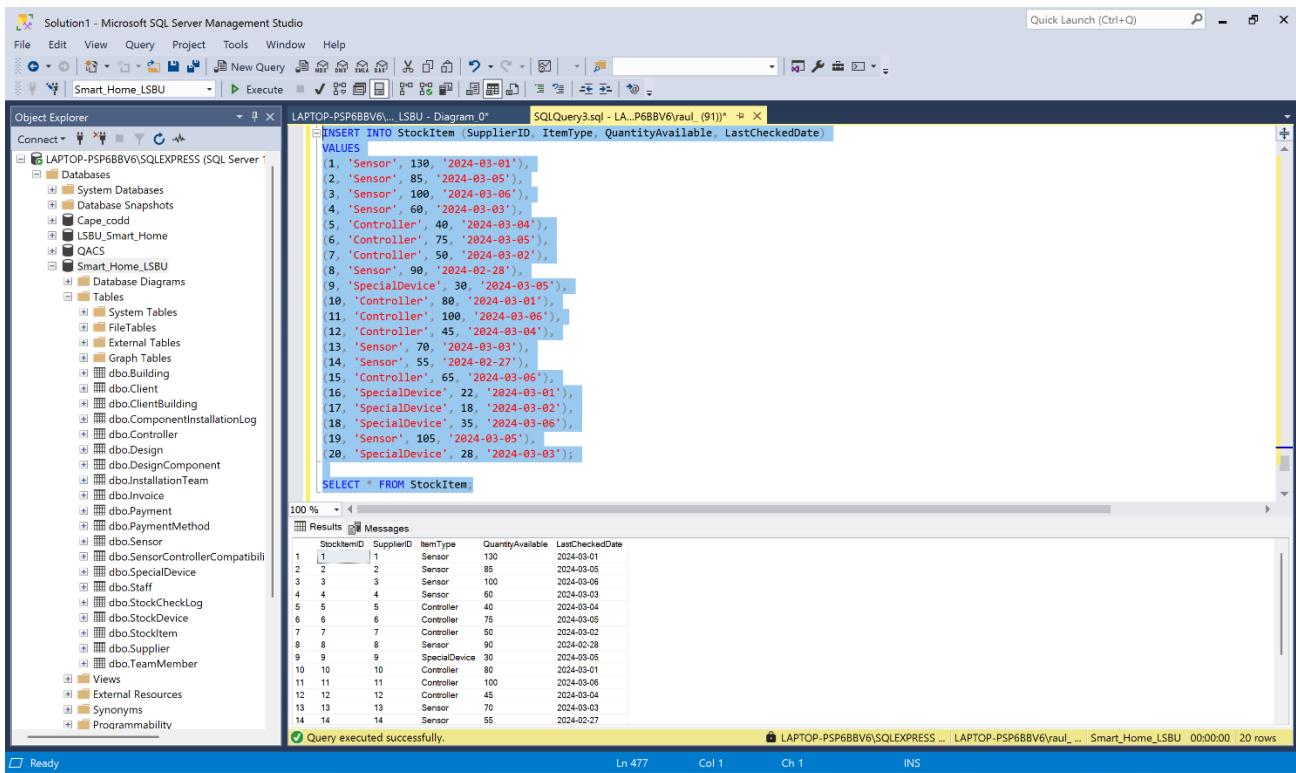
```

Results

StockItemID	SupplierID	ItemType	QuantityAvailable	LastCheckedDate	OrderStatus
1	1	WiFi Controller	130	2024-03-01	Complete
2	2	WiFi Controller	85	2024-03-05	Incomplete
3	3	WiFi Controller	100	2024-03-06	Complete
4	4	WiFi Repeater	60	2024-03-03	Cancelled
5	5	WiFi Repeater	40	2024-03-04	Incomplete

Query executed successfully.

Figure 23. Create StockItem



Solution1 - Microsoft SQL Server Management Studio

LAPTOP-PSP6BBV6\LSBU - Diagram\_0\* SQLQuery3.sql - LA...PSP6BBV6\raul\_(91)\*

```

INSERT INTO StockItem (SupplierID, ItemType, QuantityAvailable, LastCheckedDate)
VALUES
(1, 'Sensor', 130, '2024-03-01'),
(2, 'Sensor', 85, '2024-03-05'),
(3, 'Sensor', 100, '2024-03-06'),
(4, 'Sensor', 60, '2024-03-03'),
(5, 'Controller', 40, '2024-03-04'),
(6, 'Controller', 75, '2024-03-05'),
(7, 'Controller', 50, '2024-03-02'),
(8, 'Sensor', 90, '2024-02-28'),
(9, 'SpecialDevice', 30, '2024-03-05'),
(10, 'Controller', 80, '2024-03-01'),
(11, 'Controller', 100, '2024-03-06'),
(12, 'Controller', 45, '2024-03-04'),
(13, 'Sensor', 70, '2024-03-03'),
(14, 'Sensor', 55, '2024-02-27'),
(15, 'Controller', 65, '2024-03-06'),
(16, 'SpecialDevice', 22, '2024-03-01'),
(17, 'SpecialDevice', 18, '2024-03-02'),
(18, 'SpecialDevice', 35, '2024-03-06'),
(19, 'Sensor', 105, '2024-03-05'),
(20, 'SpecialDevice', 28, '2024-03-03');

SELECT * FROM StockItem;

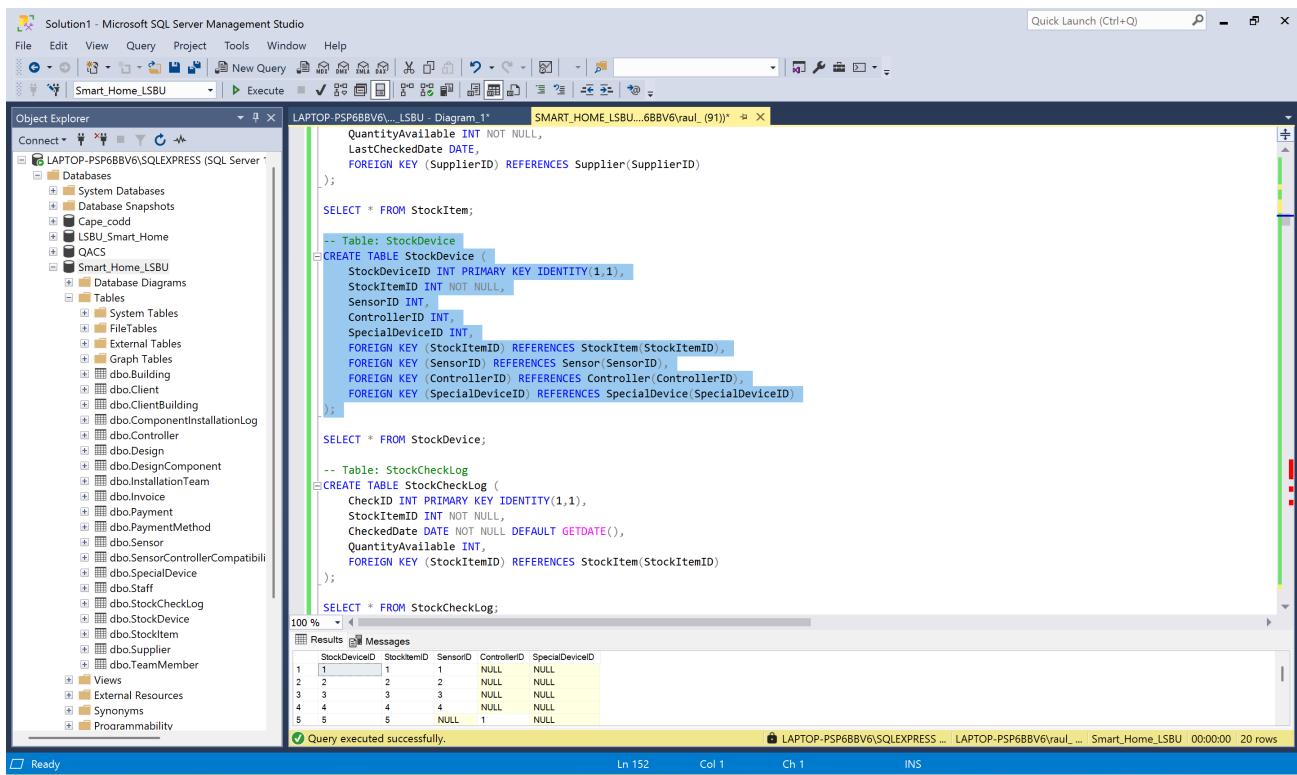
```

Results

StockItemID	SupplierID	ItemType	QuantityAvailable	LastCheckedDate
1	1	Sensor	130	2024-03-01
2	2	Sensor	85	2024-03-05
3	3	Sensor	100	2024-03-06
4	4	Sensor	60	2024-03-03
5	5	Controller	40	2024-03-04
6	6	Controller	75	2024-03-05
7	7	Controller	50	2024-03-02
8	8	Sensor	90	2024-02-28
9	9	SpecialDevice	30	2024-03-05
10	10	Controller	60	2024-03-01
11	11	Controller	100	2024-03-06
12	12	Controller	45	2024-03-04
13	13	Sensor	70	2024-03-03
14	14	Sensor	55	2024-02-27

Query executed successfully.

Figure 24. Insert Stockitem Values



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The central pane displays the SQL script for creating the 'StockDevice' table. The table has columns: StockDeviceID (primary key), StockItemID (not null), SensorID (not null), ControllerID (not null), and SpecialDeviceID (null). It includes foreign keys linking to 'StockItem', 'Sensor', 'Controller', and 'SpecialDevice'. Below the table creation, there is a select statement for 'StockDevice' and a create statement for 'StockCheckLog'. The results pane shows the execution of the select statement, returning 20 rows of data.

```

CREATE TABLE StockDevice (
    StockDeviceID INT PRIMARY KEY IDENTITY(1,1),
    StockItemID INT NOT NULL,
    SensorID INT,
    ControllerID INT,
    SpecialDeviceID INT,
    FOREIGN KEY (StockItemID) REFERENCES StockItem(StockItemID),
    FOREIGN KEY (SensorID) REFERENCES Sensor(SensorID),
    FOREIGN KEY (ControllerID) REFERENCES Controller(ControllerID),
    FOREIGN KEY (SpecialDeviceID) REFERENCES SpecialDevice(SpecialDeviceID)
);

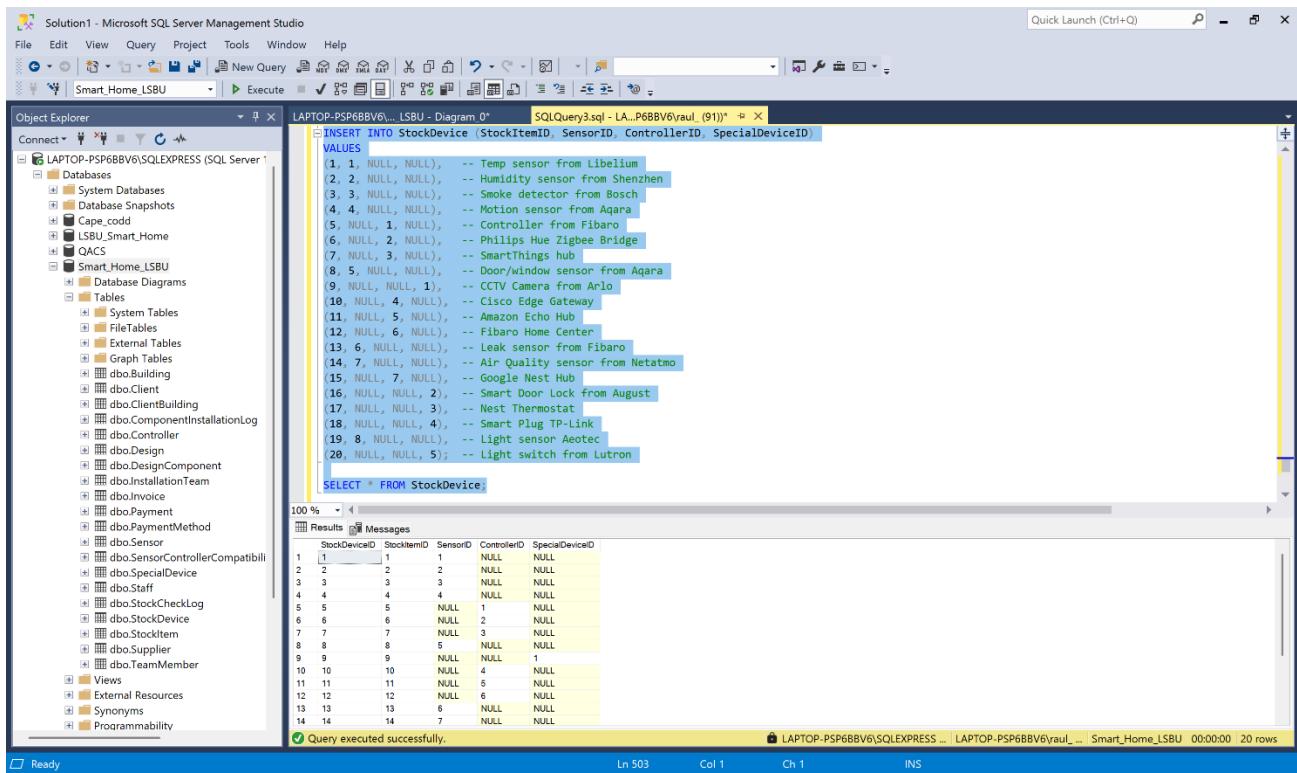
SELECT * FROM StockDevice;
-- Table: StockCheckLog
CREATE TABLE StockCheckLog (
    CheckID INT PRIMARY KEY IDENTITY(1,1),
    StockItemID INT NOT NULL,
    CheckedDate DATE NOT NULL DEFAULT GETDATE(),
    QuantityAvailable INT,
    FOREIGN KEY (StockItemID) REFERENCES StockItem(StockItemID)
);

SELECT * FROM StockCheckLog;

```

StockDeviceID	StockItemID	SensorID	ControllerID	SpecialDeviceID
1	1	1	NULL	NULL
2	2	2	NULL	NULL
3	3	3	NULL	NULL
4	4	4	NULL	NULL
5	5	NULL	1	NULL
6	6	6	NULL	2
7	7	7	NULL	3
8	8	8	NULL	4
9	9	9	NULL	NULL
10	10	10	NULL	4
11	11	11	NULL	5
12	12	12	NULL	6
13	13	13	NULL	NULL
14	14	14	NULL	NULL

Figure 25. Create StockDevice



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. The central pane displays the SQL script for inserting data into the 'StockDevice' table. The script uses a VALUES clause with 20 entries, each containing a StockDeviceID, StockItemID, SensorID, ControllerID, and SpecialDeviceID. The comments in the script describe various smart home sensors and devices. Below the insert statement, there is a select statement for 'StockDevice'. The results pane shows the execution of the insert statement, returning 20 rows of data.

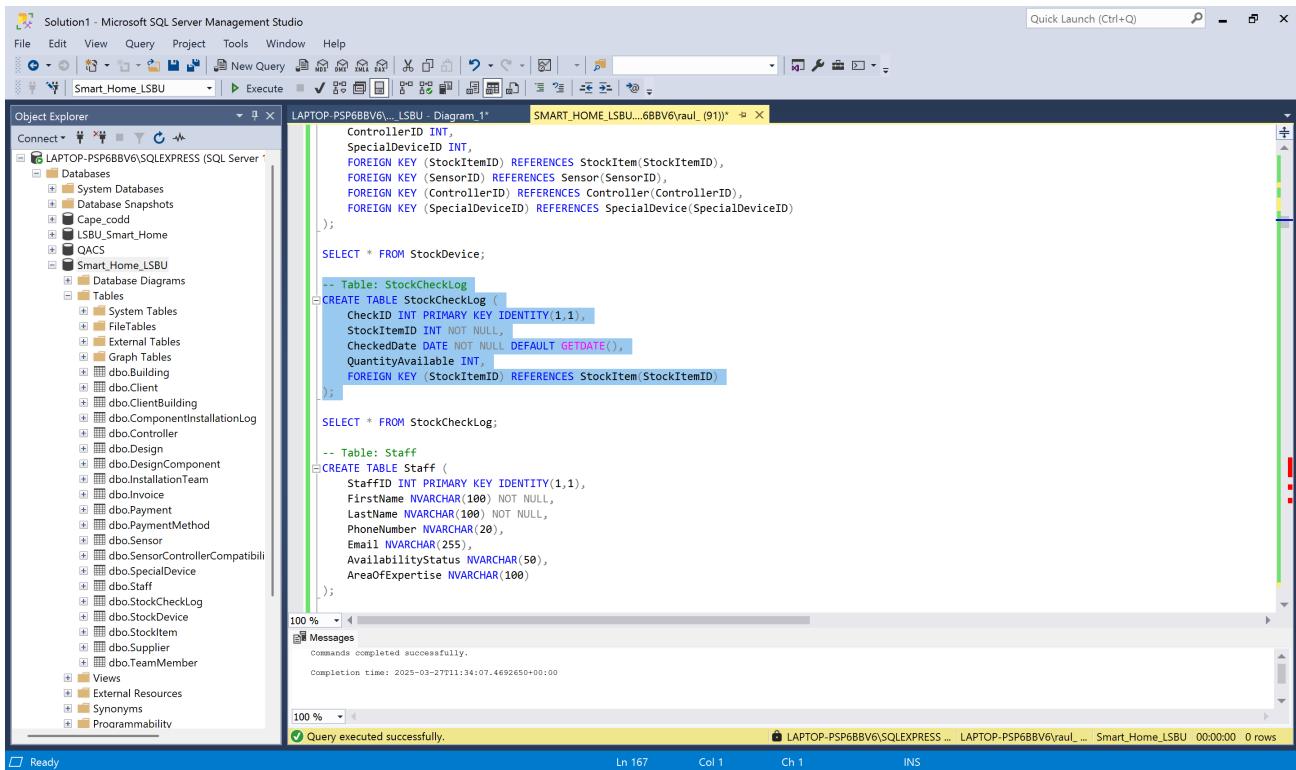
```

INSERT INTO StockDevice (StockItemID, SensorID, ControllerID, SpecialDeviceID)
VALUES
(1, 1, NULL, NULL), -- Temp sensor from Libelium
(2, 2, NULL, NULL), -- Humidity sensor from Shenzhen
(3, 3, NULL, NULL), -- Smoke detector from Bosch
(4, 4, NULL, NULL), -- Motion sensor from Aqara
(5, NULL, 1, NULL), -- Controller from Fibaro
(6, NULL, 2, NULL), -- Phillips Hue Zigbee Bridge
(7, NULL, 3, NULL), -- SmartThings hub
(8, NULL, NULL, NULL), -- Door/window sensor from Aqara
(9, NULL, NULL, 1), -- CCTV Camera from Arlo
(10, NULL, 4, NULL), -- Cisco Edge Gateway
(11, NULL, 5, NULL), -- Amazon Echo Hub
(12, NULL, 6, NULL), -- Fibaro Home Center
(13, 6, NULL, NULL), -- Leak sensor from Fibaro
(14, 7, NULL, NULL), -- Air Quality sensor from Netatmo
(15, NULL, 7, NULL), -- Google Nest Hub
(16, NULL, NULL, 2), -- Smart Door Lock from August
(17, NULL, NULL, 3), -- Nest Thermostat
(18, NULL, NULL, 4), -- Smart Plug TP-Link
(19, 8, NULL, NULL), -- Light sensor Aeotec
(20, NULL, NULL, 5), -- Light switch from Lutron

```

StockDeviceID	StockItemID	SensorID	ControllerID	SpecialDeviceID
1	1	1	NULL	NULL
2	2	2	NULL	NULL
3	3	3	NULL	NULL
4	4	4	NULL	NULL
5	5	5	NULL	1
6	6	6	NULL	2
7	7	7	NULL	3
8	8	8	NULL	4
9	9	9	NULL	NULL
10	10	10	NULL	4
11	11	11	NULL	5
12	12	12	NULL	6
13	13	13	NULL	NULL
14	14	14	NULL	NULL

Figure 26. Insert StockDevice Values



Solution1 - Microsoft SQL Server Management Studio

LAPTOP-PSP6BBV6\SQLEXPRESS (SQL Server)

Smart\_Home\_LSBU

Object Explorer

```

CREATE TABLE StockCheckLog (
    CheckID INT PRIMARY KEY IDENTITY(1,1),
    StockItemID INT NOT NULL,
    CheckedDate DATE NOT NULL DEFAULT GETDATE(),
    QuantityAvailable INT,
    FOREIGN KEY (StockItemID) REFERENCES StockItem(StockItemID)
);

```

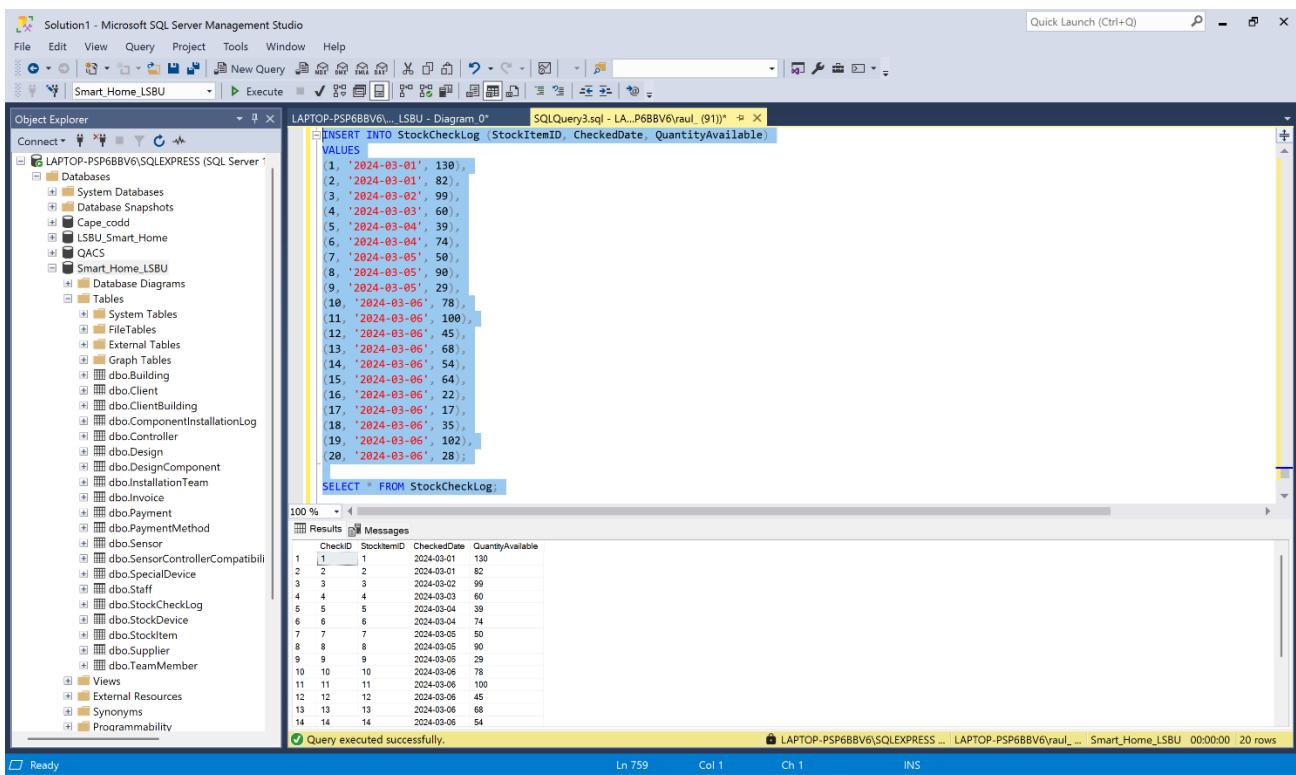
Messages

Commands completed successfully.

Completion time: 2023-03-27T11:34:07.469Z 0 rows

Query executed successfully.

Figure 27. Create StockCheckLog



Solution1 - Microsoft SQL Server Management Studio

LAPTOP-PSP6BBV6\SQLEXPRESS (SQL Server)

Smart\_Home\_LSBU

Object Explorer

```

INSERT INTO StockCheckLog (StockItemID, CheckedDate, QuantityAvailable)
VALUES
(1, '2024-03-01', 130),
(2, '2024-03-01', 82),
(3, '2024-03-02', 99),
(4, '2024-03-03', 60),
(5, '2024-03-04', 39),
(6, '2024-03-04', 74),
(7, '2024-03-05', 50),
(8, '2024-03-05', 99),
(9, '2024-03-05', 29),
(10, '2024-03-06', 78),
(11, '2024-03-06', 100),
(12, '2024-03-06', 45),
(13, '2024-03-06', 68),
(14, '2024-03-06', 54),
(15, '2024-03-06', 64),
(16, '2024-03-06', 22),
(17, '2024-03-06', 17),
(18, '2024-03-06', 35),
(19, '2024-03-06', 102),
(20, '2024-03-06', 28);

```

Results

CheckID	StockItemID	CheckedDate	QuantityAvailable
1	1	2024-03-01	130
2	2	2024-03-01	82
3	3	2024-03-02	99
4	4	2024-03-03	60
5	5	2024-03-04	39
6	6	2024-03-04	74
7	7	2024-03-05	50
8	8	2024-03-05	99
9	9	2024-03-05	29
10	10	2024-03-06	78
11	11	2024-03-06	100
12	12	2024-03-06	45
13	13	2024-03-06	68
14	14	2024-03-06	54

Messages

Query executed successfully.

Figure 28. Insert StockCheckLog Values

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. A query window titled 'LAPTOP-PSP6BBV6\LSBU - Diagram.1\*' contains the following SQL code:

```

CREATE TABLE Staff (
    StaffID INT PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(100) NOT NULL,
    LastName NVARCHAR(100) NOT NULL,
    PhoneNumber NVARCHAR(20),
    Email NVARCHAR(255),
    AvailabilityStatus NVARCHAR(50),
    AreaOfExpertise NVARCHAR(100)
);

SELECT * FROM Staff;

```

The results pane shows a table with 5 rows of sample data:

StaffID	FirstName	LastName	PhoneNumber	Email	AvailabilityStatus	AreaOfExpertise
1	James	Walker	07300111223	james.walker@lsbu-smarthome.co.uk	Available	Sensor Installation
2	Amelia	Scott	07400987651	amelia.scott@lsbu-smarthome.co.uk	On Leave	Electrical Engineering
3	Liam	Thompson	07500123887	liam.thompson@lsbu-smarthome.co.uk	Available	System Integration
4	Olivia	White	07600456789	olivia.white@lsbu-smarthome.co.uk	On Site	CCTV Installation
5	Noah	Morris	07344556677	noah.morris@lsbu-smarthome.co.uk	Available	Controller Configuration

A status bar at the bottom indicates 'Query executed successfully.'

Figure 29. Create Staff

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home LSBU'. A query window titled 'LAPTOP-PSP6BBV6\LSBU - Diagram.0\*' contains the following SQL code:

```

INSERT INTO Staff (FirstName, LastName, PhoneNumber, Email, AvailabilityStatus, AreaOfExpertise)
VALUES
('James', 'Walker', '07300111223', 'james.walker@lsbu-smarthome.co.uk', 'Available', 'Sensor Installation'),
('Amelia', 'Scott', '07400987651', 'amelia.scott@lsbu-smarthome.co.uk', 'On Leave', 'Electrical Engineering'),
('Liam', 'Thompson', '07500123887', 'liam.thompson@lsbu-smarthome.co.uk', 'Available', 'System Integration'),
('Olivia', 'White', '07600456789', 'olivia.white@lsbu-smarthome.co.uk', 'On Site', 'CCTV Installation'),
('Noah', 'Morris', '07344556677', 'noah.morris@lsbu-smarthome.co.uk', 'Available', 'Controller Configuration'),
('Isla', 'Evans', '07588990011', 'isla.evans@lsbu-smarthome.co.uk', 'On Site', 'HVAC Integration'),
('William', 'Baker', '07766543210', 'william.baker@lsbu-smarthome.co.uk', 'Available', 'Data Networking'),
('Ava', 'Cooper', '07333445666', 'ava.cooper@lsbu-smarthome.co.uk', 'On Leave', 'Fire Safety Systems'),
('Lucas', 'Hall', '07455443322', 'lucas.hall@lsbu-smarthome.co.uk', 'Available', 'Smart Device Pairing'),
('Freya', 'Morgan', '07312233445', 'freya.morgan@lsbu-smarthome.co.uk', 'On Site', 'Mobile App Testing'),
('Henry', 'Ward', '07599887766', 'henry.ward@lsbu-smarthome.co.uk', 'Available', 'Lighting Automation'),
('Mia', 'Young', '07422113344', 'mia.young@lsbu-smarthome.co.uk', 'Available', 'Customer Support'),
('George', 'Turner', '07333557788', 'george.turner@lsbu-smarthome.co.uk', 'On Leave', 'Sensor Tuning'),
('Lily', 'Parker', '07477889900', 'lily.parker@lsbu-smarthome.co.uk', 'Available', 'Energy Systems'),
('Charlie', 'Reed', '07600011234', 'charlie.reed@lsbu-smarthome.co.uk', 'On Site', 'Wiring & Cabling'),
('Grace', 'James', '07522334455', 'grace.james@lsbu-smarthome.co.uk', 'Available', 'Client Training'),
('Leo', 'Bennett', '07388766555', 'leo.bennett@lsbu-smarthome.co.uk', 'On Site', 'Diagnostics & Repairs'),
('Sophie', 'Murphy', '07466554433', 'sophie.murphy@lsbu-smarthome.co.uk', 'Available', 'IoT Firmware'),
('Jacob', 'Ross', '07544433221', 'jacob.ross@lsbu-smarthome.co.uk', 'Available', 'Automation Configuration'),
('Emily', 'Campbell', '07322119988', 'emily.campbell@lsbu-smarthome.co.uk', 'On Leave', 'Security Systems');

SELECT * FROM Staff;

```

The results pane shows a table with 14 rows of sample data, identical to Figure 29.

A status bar at the bottom indicates 'Query executed successfully.'

Figure 30. Insert Staff Values

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home LSBU'. The central pane displays a T-SQL script for creating the 'InstallationTeam' table and inserting data into it. The results pane shows the successful execution of the query and the resulting data.

```

SELECT * FROM Staff;

-- Table: InstallationTeam
CREATE TABLE InstallationTeam (
    TeamID INT PRIMARY KEY IDENTITY(1,1),
    TeamName NVARCHAR(100)
);

SELECT * FROM InstallationTeam;

-- Table: TeamMember
CREATE TABLE TeamMember (
    TeamID INT NOT NULL,
    StaffID INT NOT NULL,
    PRIMARY KEY (TeamID, StaffID),
    FOREIGN KEY (TeamID) REFERENCES InstallationTeam(TeamID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

SELECT * FROM TeamMember;

-- Table: ComponentInstallationLog
CREATE TABLE ComponentInstallationLog (
    ComponentID INT NOT NULL,
    StaffID INT NOT NULL,
    InstallDate DATE NOT NULL DEFAULT GETDATE(),
    Notes NVARCHAR(255),
    PRIMARY KEY (ComponentID, StaffID),
    FOREIGN KEY (ComponentID) REFERENCES Component(ComponentID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

```

TeamID	TeamName
1	Team Alpha
2	Team Beta
3	Team Gamma
4	Team Delta
5	Team Epsilon

Query executed successfully.

Figure 31. Create InstallationTeam

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home LSBU'. The central pane displays a T-SQL script for inserting data into the 'InstallationTeam' table. The results pane shows the successful execution of the query and the resulting data.

```

-- Add some sample data
INSERT INTO Staff (StaffID, FirstName, LastName, Email, Available, Location)
VALUES ('1', 'John', 'Doe', 'jdoe@lsbu-smarthome.co.uk', 'Available', 'Client Training'),
       ('2', 'Leo', 'Bennett', 'l.bennett@lsbu-smarthome.co.uk', 'On Site', 'Diagnostics & Repairs'),
       ('3', 'Sophie', 'Murray', 'sophie.murray@lsbu-smarthome.co.uk', 'Available', 'IoT Firmware'),
       ('4', 'Jacob', 'Ross', '07544433221', 'jacob.ross@lsbu-smarthome.co.uk', 'Available', 'Automation Configuration'),
       ('5', 'Emily', 'Campbell', 'emily.campbell@lsbu-smarthome.co.uk', 'On Leave', 'Security Systems');

SELECT * FROM Staff;

--create the Installation Teams and assign staff to them using the InstallationTeam and TeamMember tables--

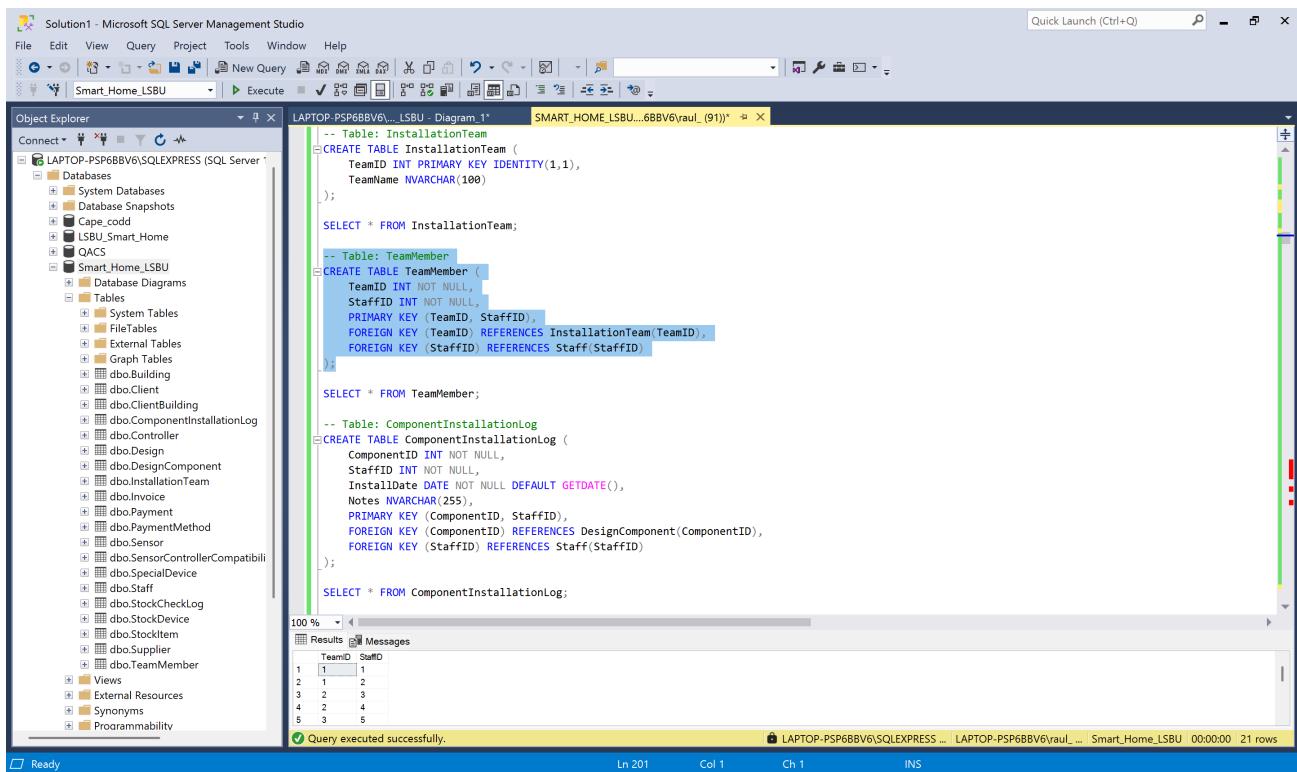
-- Inserting into InstallationTeam
INSERT INTO InstallationTeam (TeamName)
VALUES ('Team Alpha'), ('Team Beta'), ('Team Gamma'), ('Team Delta'), ('Team Epsilon'), ('Team Zeta'), ('Team Theta'), ('Team Omega'), ('Team Sigma'), ('Team Nova');

```

TeamID	TeamName
1	Team Alpha
2	Team Beta
3	Team Gamma
4	Team Delta
5	Team Epsilon
6	Team Zeta
7	Team Theta
8	Team Omega
9	Team Sigma
10	Team Nova

Query executed successfully.

Figure 32. Insert InstallationTeam Values



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home\_LSBU

Object Explorer

```
-- Table: InstallationTeam
CREATE TABLE InstallationTeam (
    TeamID INT PRIMARY KEY IDENTITY(1,1),
    TeamName NVARCHAR(100)
);

-- Table: TeamMember
CREATE TABLE TeamMember (
    TeamID INT NOT NULL,
    StaffID INT NOT NULL,
    PRIMARY KEY (TeamID, StaffID),
    FOREIGN KEY (TeamID) REFERENCES InstallationTeam(TeamID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

-- Table: ComponentInstallationLog
CREATE TABLE ComponentInstallationLog (
    ComponentID INT NOT NULL,
    StaffID INT NOT NULL,
    InstallDate DATE NOT NULL DEFAULT GETDATE(),
    Notes NVARCHAR(255),
    PRIMARY KEY (ComponentID, StaffID),
    FOREIGN KEY (ComponentID) REFERENCES DesignComponent(ComponentID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

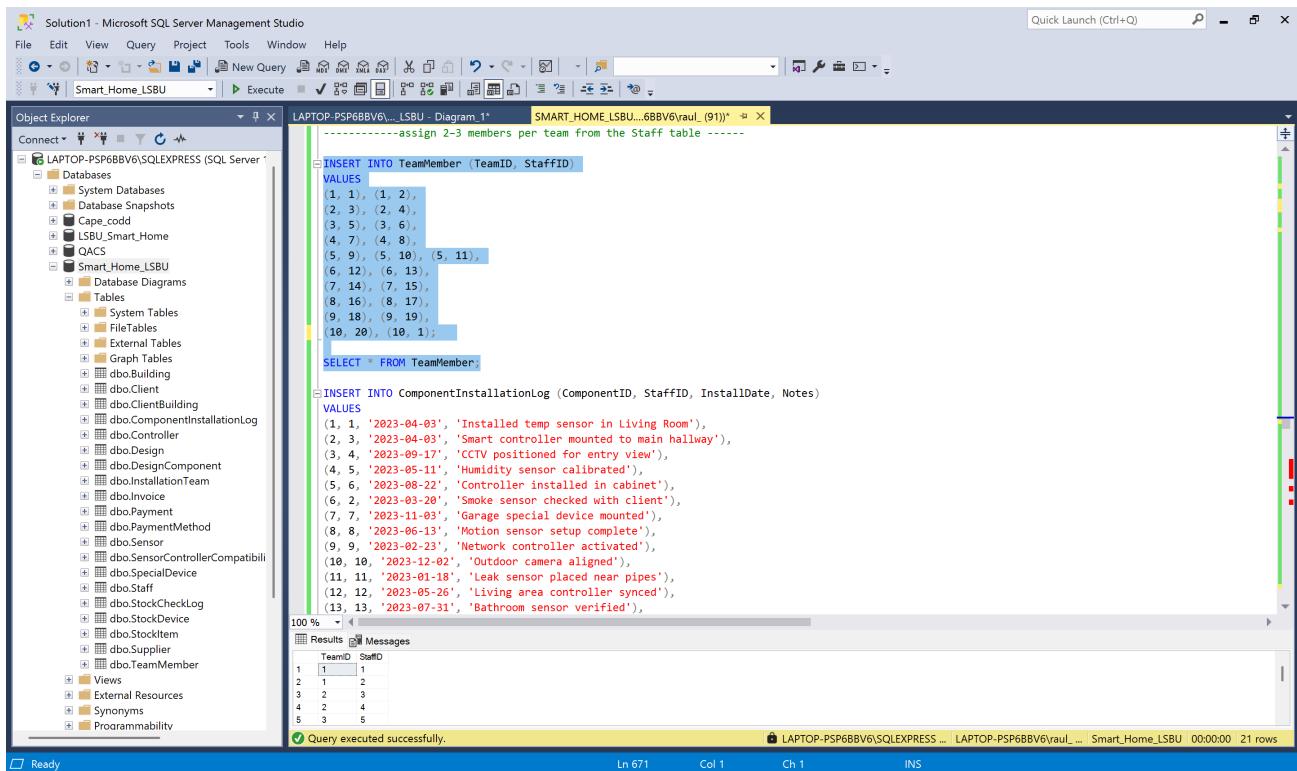
SELECT * FROM ComponentInstallationLog;
```

Results

TeamID	StaffID
1	1
2	1
3	2
4	2
5	3

Query executed successfully.

Figure 33. Create TeamMember



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home\_LSBU

Object Explorer

```
-- assign 2-3 members per team from the Staff table -----
INSERT INTO TeamMember (TeamID, StaffID)
VALUES
(1, 1), (1, 2),
(2, 3), (2, 4),
(3, 5), (3, 6),
(4, 7), (4, 8),
(5, 9), (5, 10), (5, 11),
(6, 12), (6, 13),
(7, 14), (7, 15),
(8, 16), (8, 17),
(9, 18), (9, 19),
(10, 20), (10, 1);

SELECT * FROM TeamMember;

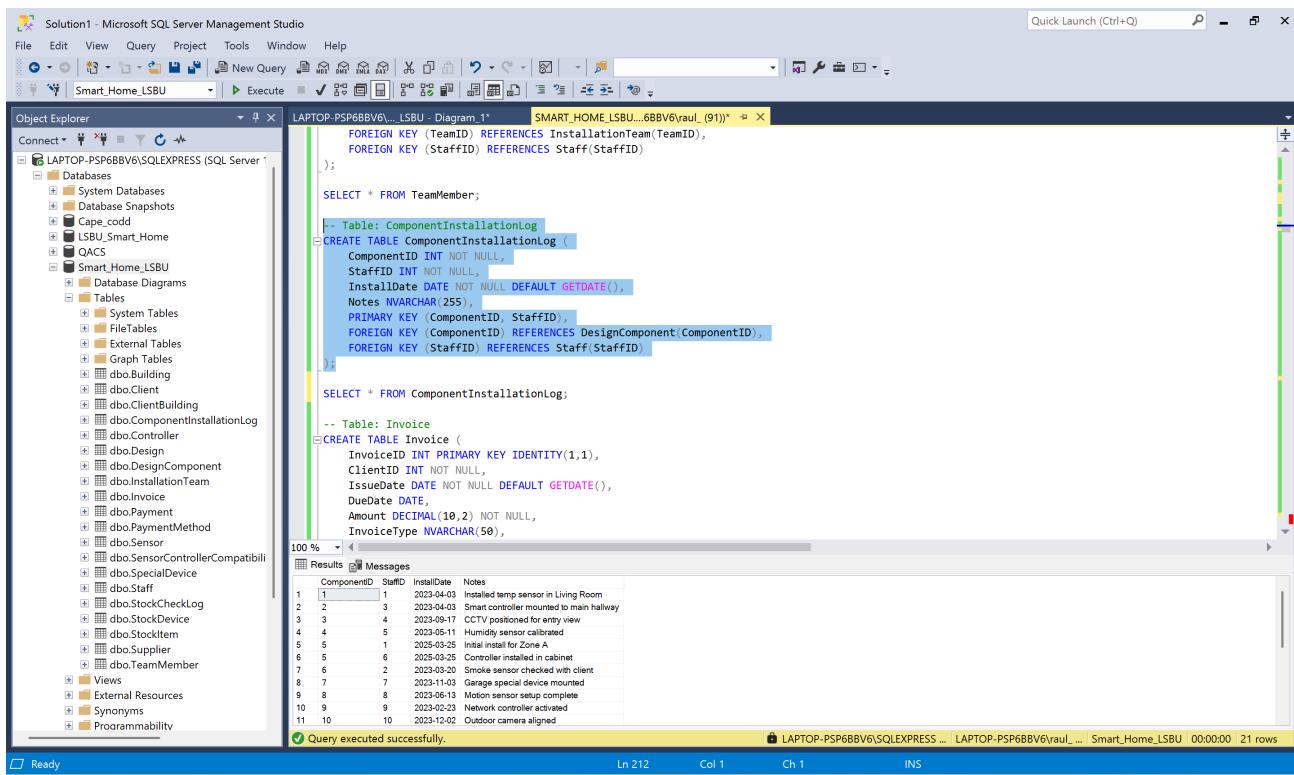
-- INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
VALUES
(1, 1, '2023-04-03', 'Installed temp sensor in Living Room'),
(2, 3, '2023-04-03', 'Smart controller mounted to main hallway'),
(3, 4, '2023-09-17', 'CCTV positioned for entry view'),
(4, 5, '2023-05-11', 'Humidity sensor calibrated'),
(5, 6, '2023-08-22', 'Controller installed in cabinet'),
(6, 2, '2023-03-20', 'Smoke sensor checked with client'),
(7, 7, '2023-11-03', 'Garage special device mounted'),
(8, 8, '2023-06-13', 'Motion sensor setup complete'),
(9, 9, '2023-02-23', 'Network controller activated'),
(10, 10, '2023-12-02', 'Outdoor camera aligned'),
(11, 11, '2023-01-18', 'Leak sensor placed near pipes'),
(12, 12, '2023-05-26', 'Living area controller synced'),
(13, 13, '2023-07-31', 'Bathroom sensor verified');
```

Results

TeamID	StaffID
1	1
2	1
3	2
4	2
5	3

Query executed successfully.

Figure 34. Insert TeamMember Values



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. The main pane displays the creation script for the 'ComponentInstallationLog' table:

```

CREATE TABLE ComponentInstallationLog (
    ComponentID INT NOT NULL,
    StaffID INT NOT NULL,
    InstallDate DATE NOT NULL DEFAULT GETDATE(),
    Notes NVARCHAR(255),
    PRIMARY KEY (ComponentID, StaffID),
    FOREIGN KEY (ComponentID) REFERENCES DesignComponent(ComponentID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

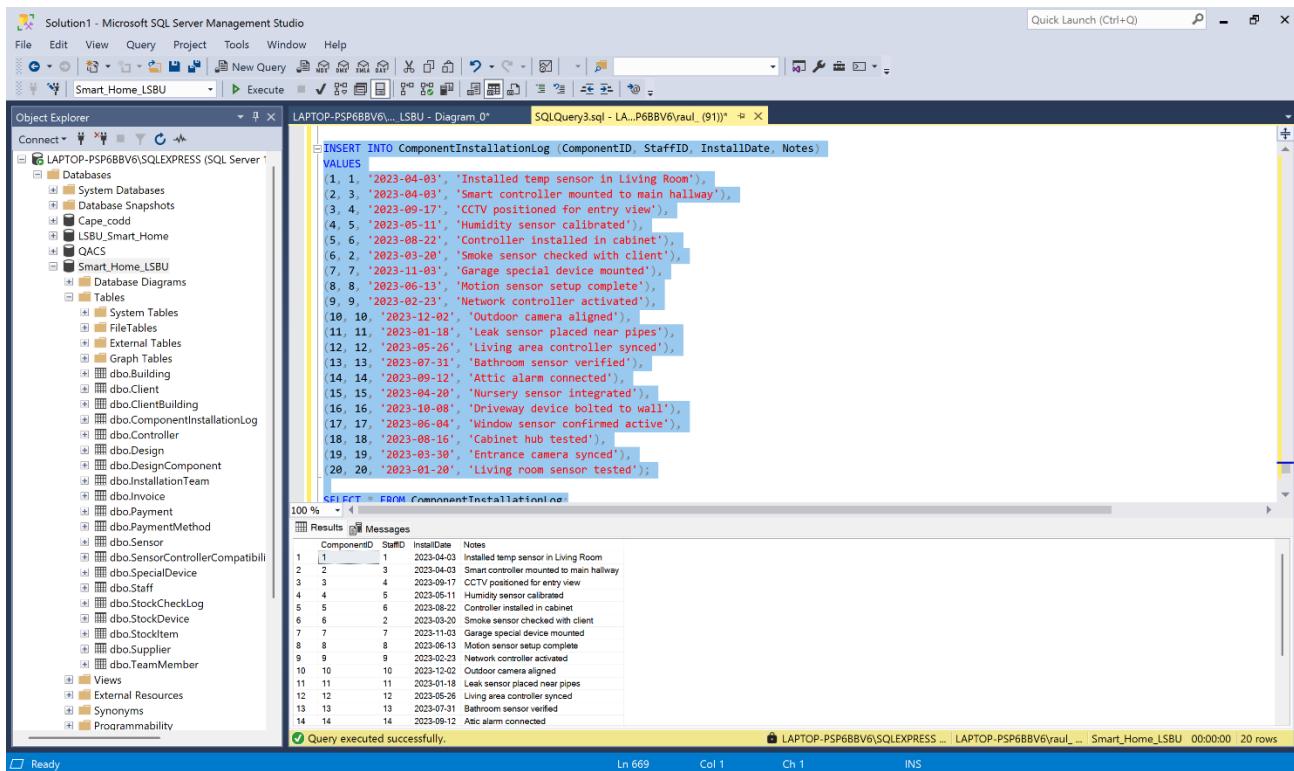
```

Below the script, the results pane shows the output of a SELECT query on the newly created table:

ComponentID	StaffID	InstallDate	Notes
1	1	2023-04-03	Installed temp sensor in Living Room
2	2	2023-04-03	Smart controller mounted to main hallway
3	3	2023-09-17	CCTV positioned for entry view
4	4	2023-05-11	Humidity sensor calibrated
5	5	2023-03-25	Initial install for Zone A
6	6	2023-03-25	Controller installed in cabinet
7	6	2023-03-20	Smoke sensor checked with client
8	7	2023-11-03	Garage special device mounted
9	8	2023-06-13	Motion sensor setup complete
10	9	2023-02-23	Network controller activated
11	10	2023-12-02	Outdoor camera aligned

A status bar at the bottom indicates 'Query executed successfully.'

Figure 35. Create ComponentInstallationLog



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. The main pane displays the execution of an INSERT INTO statement into the 'ComponentInstallationLog' table:

```

INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
VALUES
    (1, 1, '2023-04-03', 'Installed temp sensor in Living Room'),
    (2, 3, '2023-04-03', 'Smart controller mounted to main hallway'),
    (3, 4, '2023-09-17', 'CCTV positioned for entry view'),
    (4, 5, '2023-05-11', 'Humidity sensor calibrated'),
    (5, 6, '2023-08-22', 'Controller installed in cabinet'),
    (6, 2, '2023-03-20', 'Smoke sensor checked with client'),
    (7, 7, '2023-11-03', 'Garage special device mounted'),
    (8, 8, '2023-06-13', 'Motion sensor setup complete'),
    (9, 9, '2023-02-23', 'Network controller activated'),
    (10, 10, '2023-12-02', 'Outdoor camera aligned'),
    (11, 11, '2023-01-18', 'Leak sensor placed near pipes'),
    (12, 12, '2023-05-26', 'Living area controller synced'),
    (13, 13, '2023-07-31', 'Bathroom sensor verified'),
    (14, 14, '2023-09-12', 'Attic alarm connected'),
    (15, 15, '2023-04-20', 'Nursery sensor integrated'),
    (16, 16, '2023-10-08', 'Driveway device bolted to wall'),
    (17, 17, '2023-06-04', 'Window sensor confirmed active'),
    (18, 18, '2023-08-16', 'Cabinet hub tested'),
    (19, 19, '2023-03-30', 'Entrance camera synced'),
    (20, 20, '2023-01-20', 'Living room sensor tested');

```

Below the script, the results pane shows the output of a SELECT query on the table:

ComponentID	StaffID	InstallDate	Notes
1	1	2023-04-03	Installed temp sensor in Living Room
2	3	2023-04-03	Smart controller mounted to main hallway
3	4	2023-09-17	CCTV positioned for entry view
4	5	2023-05-11	Humidity sensor calibrated
5	6	2023-08-22	Controller installed in cabinet
6	2	2023-03-20	Smoke sensor checked with client
7	7	2023-11-03	Garage special device mounted
8	8	2023-06-13	Motion sensor setup complete
9	9	2023-02-23	Network controller activated
10	10	2023-12-02	Outdoor camera aligned
11	11	2023-01-18	Leak sensor placed near pipes
12	12	2023-05-26	Living area controller synced
13	13	2023-07-31	Bathroom sensor verified
14	14	2023-09-12	Attic alarm connected

A status bar at the bottom indicates 'Query executed successfully.'

Figure 36. Insert ComponentInstallationLog Values

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'Smart\_Home LSBU' is selected. In the center pane, a new query window titled 'LAPTOP-PSP6BBV6...\_LSBU - Diagram.1\*' is open, displaying the creation script for the 'Invoice' table. The script includes primary key constraints, foreign key references to 'DesignComponent' and 'Staff', and a table creation statement with columns: InvoiceID (INT, primary key identity(1,1)), ClientID (INT, not null), IssueDate (DATE, not null, default GETDATE()), DueDate (DATE), Amount (DECIMAL(10, 2), not null), and InvoiceType (NVARCHAR(50)). A foreign key constraint 'FK\_Invoice\_Client' is also defined. Below the table creation, a select statement is shown to retrieve all columns from the 'Invoice' table.

```

PRIMARY KEY (ComponentID, StaffID),
FOREIGN KEY (ComponentID) REFERENCES DesignComponent(ComponentID),
FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
);

-- Table: Invoice
CREATE TABLE Invoice (
    InvoiceID INT PRIMARY KEY IDENTITY(1,1),
    ClientID INT NOT NULL,
    IssueDate DATE NOT NULL DEFAULT GETDATE(),
    DueDate DATE,
    Amount DECIMAL(10, 2) NOT NULL,
    InvoiceType NVARCHAR(50),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

SELECT * FROM Invoice;

```

Figure 37. create Invoice

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'Smart\_Home LSBU' is selected. In the center pane, a new query window titled 'LAPTOP-PSP6BBV6...\_LSBU - Diagram.0\*' is open, displaying an 'INSERT INTO' statement for the 'Invoice' table. The statement inserts 20 rows of data with columns: ClientID, IssueDate, DueDate, Amount, and InvoiceType. The data spans from 2023-04-05 to 2023-02-18, with various amounts and types (Design, Installation, Maintenance). Below the insert statement, a select statement is shown to retrieve all columns from the 'Invoice' table.

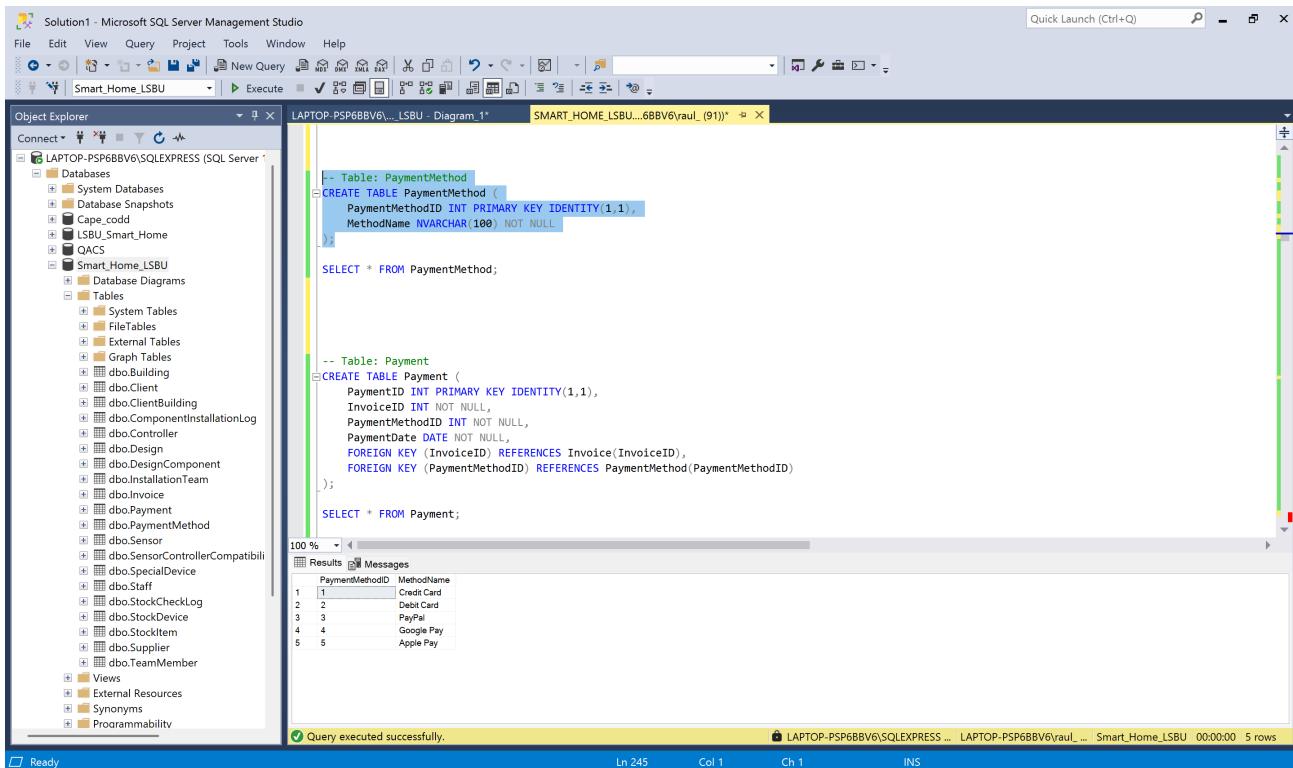
```

VALUES
(1, '2023-04-05', '2023-05-03', 850.00, 'Design'),
(2, '2023-05-12', '2023-06-09', 1200.00, 'Installation'),
(3, '2023-07-08', '2023-08-05', 300.00, 'Maintenance'),
(4, '2023-09-02', '2023-09-30', 975.00, 'Design'),
(5, '2023-03-20', '2023-04-17', 1500.00, 'Installation'),
(6, '2023-11-15', '2023-12-13', 400.00, 'Maintenance'),
(7, '2023-06-10', '2023-07-08', 2100.00, 'Installation'),
(8, '2023-02-14', '2023-03-13', 750.00, 'Design'),
(9, '2023-12-01', '2023-12-28', 1300.00, 'Installation'),
(10, '2023-01-10', '2023-02-07', 250.00, 'Maintenance'),
(11, '2023-05-26', '2023-06-23', 990.00, 'Design')

SELECT * FROM Invoice;

```

Figure 38. Insert Invoice Values



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home LSBU Diagram\_1\*

```
-- Table: PaymentMethod
CREATE TABLE PaymentMethod (
    PaymentMethodID INT PRIMARY KEY IDENTITY(1,1),
    MethodName NVARCHAR(100) NOT NULL
);

SELECT * FROM PaymentMethod;

-- Table: Payment
CREATE TABLE Payment (
    PaymentID INT PRIMARY KEY IDENTITY(1,1),
    InvoiceID INT NOT NULL,
    PaymentMethodID INT NOT NULL,
    PaymentDate DATE NOT NULL,
    FOREIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceID),
    FOREIGN KEY (PaymentMethodID) REFERENCES PaymentMethod(PaymentMethodID)
);

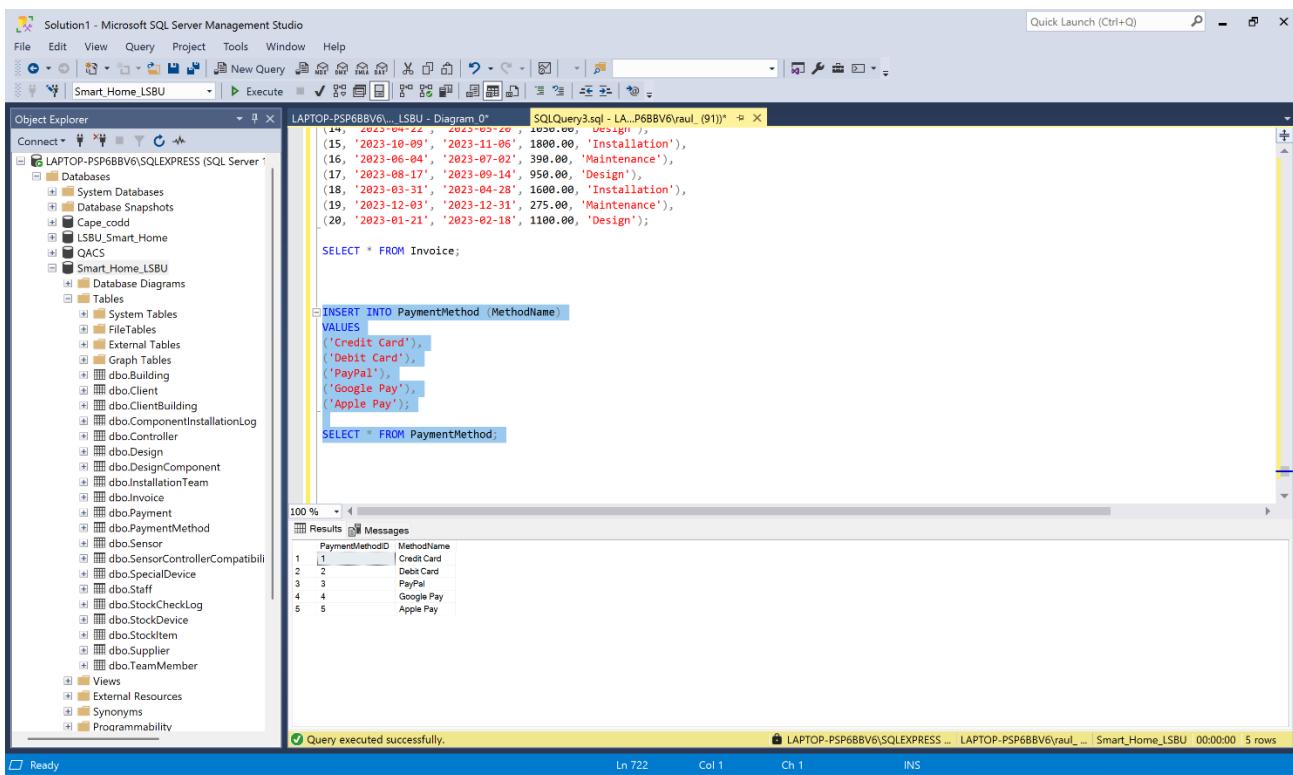
SELECT * FROM Payment;
```

Result Messages

PaymentMethodID	MethodName
1	Credit Card
2	Debit Card
3	PayPal
4	Google Pay
5	Apple Pay

Query executed successfully.

Figure 39. Create PaymentMethod



Solution1 - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Smart\_Home LSBU Diagram 0\*

```
(14, '2023-04-22', '2023-03-20', 1000.00, 'Design'),
(15, '2023-10-09', '2023-11-06', 1800.00, 'Installation'),
(16, '2023-06-04', '2023-07-02', 300.00, 'Maintenance'),
(17, '2023-08-17', '2023-09-14', 950.00, 'Design'),
(18, '2023-03-31', '2023-04-28', 1600.00, 'Installation'),
(19, '2023-12-03', '2023-12-31', 275.00, 'Maintenance'),
(20, '2023-01-21', '2023-02-18', 1100.00, 'Design');

SELECT * FROM Invoice;

INSERT INTO PaymentMethod (MethodName)
VALUES
('Credit Card'),
('Debit Card'),
('PayPal'),
('Google Pay'),
('Apple Pay');

SELECT * FROM PaymentMethod;
```

Result Messages

PaymentMethodID	MethodName
1	Credit Card
2	Debit Card
3	PayPal
4	Google Pay
5	Apple Pay

Query executed successfully.

Figure 40. Insert PaymentMethod Values

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home LSBU'. The central pane displays the following T-SQL code:

```

-- Table: Payment
CREATE TABLE Payment (
    PaymentID INT PRIMARY KEY IDENTITY(1,1),
    InvoiceID INT NOT NULL,
    PaymentMethodID INT NOT NULL,
    PaymentDate DATE NOT NULL,
    FOREIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceID),
    FOREIGN KEY (PaymentMethodID) REFERENCES PaymentMethod(PaymentMethodID)
);

SELECT * FROM Payment;

-- SELECT name AS TableName
-- FROM sys.tables
-- ORDER BY name;

```

The 'Messages' tab at the bottom shows the output: 'Query executed successfully.'

Figure 41. Create Payment

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'Smart\_Home LSBU'. The central pane displays the following T-SQL code:

```

-- Payment Method
('PayPal'),
('Google Pay'),
('Apple Pay');

SELECT * FROM PaymentMethod;

INSERT INTO Payment (InvoiceID, PaymentMethodID, PaymentDate)
VALUES
(1, 1, '2023-04-20'),
(2, 2, '2023-05-25'),
(3, 3, '2023-07-25'),
(4, 4, '2023-09-25'),
(5, 1, '2023-04-12'),
(6, 2, '2023-11-30'),
(7, 5, '2023-06-25'),
(8, 3, '2023-03-02'),
(9, 1, '2023-12-15'),
(10, 4, '2023-01-25'),
(11, 2, '2023-06-10'),
(12, 1, '2023-08-12'),
(13, 5, '2023-09-27'),
(14, 3, '2023-05-01'),
(15, 2, '2023-10-20'),
(16, 1, '2023-06-25'),
(17, 4, '2023-09-02');

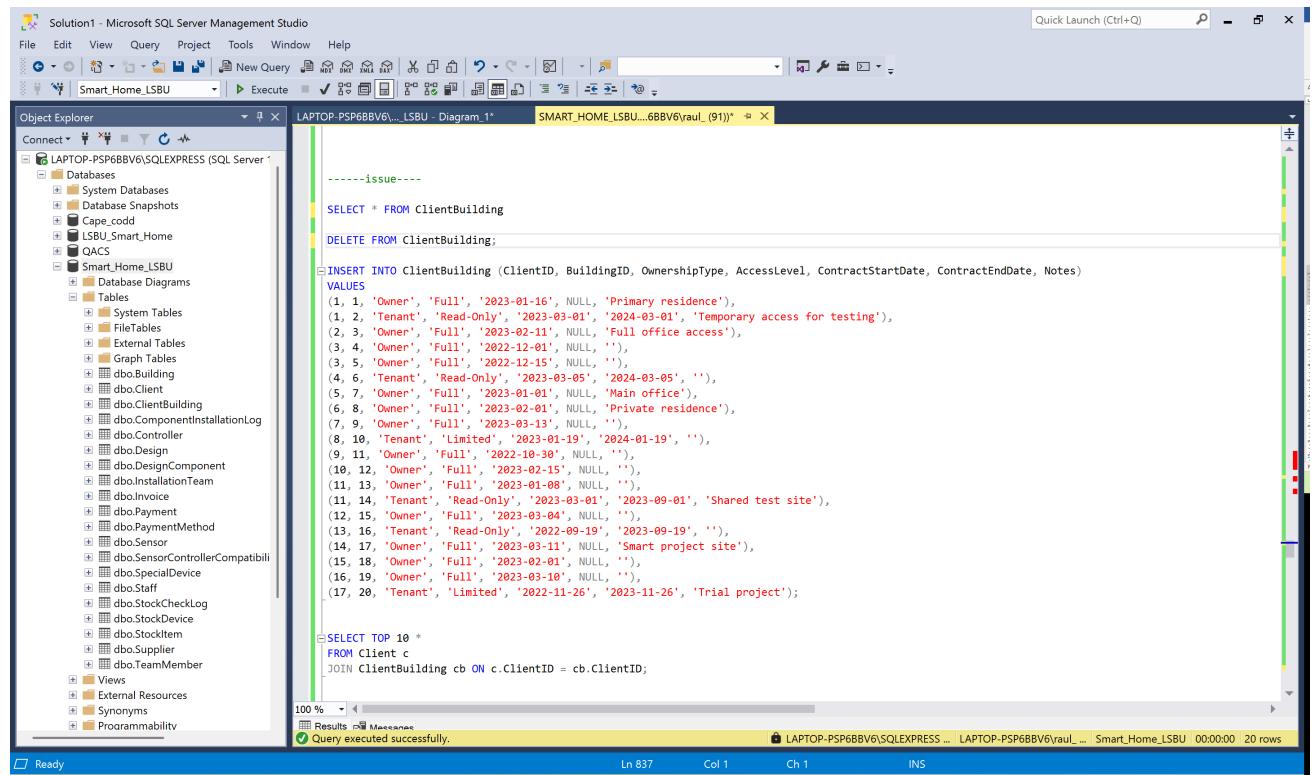
```

The 'Messages' tab at the bottom shows the output: 'Query executed successfully.'

Figure 42. Insert Payment Values

The problem with the ClientBuilding table was that the ClientID values didn't match those in the Client table. This meant that when trying to join the two tables, no data appeared because there were no matching IDs. Once we deleted the incorrect records and reinserted them using valid ClientIDs, the join worked correctly, and the relationships were restored.

This can be observed in *Figure 43*.



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists databases, tables, and other objects under the 'Smart\_Home\_LSBU' database. The central pane displays a query script titled 'issue---'. The script starts with a comment '-----issue----', followed by a 'SELECT \* FROM ClientBuilding' statement, a 'DELETE FROM ClientBuilding' statement, an 'INSERT INTO ClientBuilding' statement with 17 rows of data, and a 'SELECT TOP 10 \*' statement. The status bar at the bottom right indicates 'Query executed successfully.' and '20 rows'.

```

-----issue-----
SELECT * FROM ClientBuilding
DELETE FROM ClientBuilding;
INSERT INTO ClientBuilding (ClientID, BuildingID, OwnershipType, AccessLevel, ContractStartDate, ContractEndDate, Notes)
VALUES
(1, 1, 'Owner', 'Full', '2023-01-16', NULL, 'Primary residence'),
(1, 2, 'Tenant', 'Read-Only', '2023-03-01', '2024-03-01', 'Temporary access for testing'),
(2, 3, 'Owner', 'Full', '2023-02-11', NULL, 'Full office access'),
(3, 4, 'Owner', 'Full', '2022-12-01', NULL, ''),
(3, 5, 'Owner', 'Full', '2022-12-15', NULL, ''),
(4, 6, 'Tenant', 'Read-Only', '2023-03-05', '2024-03-05', ''),
(5, 7, 'Owner', 'Full', '2023-01-01', NULL, 'Main office'),
(6, 8, 'Owner', 'Full', '2023-02-01', NULL, 'Private residence'),
(7, 9, 'Owner', 'Full', '2023-03-13', NULL, ''),
(8, 10, 'Tenant', 'Limited', '2023-01-19', '2024-01-19', ''),
(9, 11, 'Owner', 'Full', '2022-10-30', NULL, ''),
(10, 12, 'Owner', 'Full', '2023-02-15', NULL, ''),
(11, 13, 'Owner', 'Full', '2023-01-08', NULL, ''),
(11, 14, 'Tenant', 'Read-Only', '2023-03-01', '2023-09-01', 'Shared test site'),
(12, 15, 'Owner', 'Full', '2023-03-04', NULL, ''),
(13, 16, 'Tenant', 'Read-Only', '2022-09-19', '2023-09-19', ''),
(14, 17, 'Owner', 'Full', '2023-03-11', NULL, 'Smart project site'),
(15, 18, 'Owner', 'Full', '2023-02-01', NULL, ''),
(16, 19, 'Owner', 'Full', '2023-03-10', NULL, ''),
(17, 20, 'Tenant', 'Limited', '2022-11-26', '2023-11-26', 'Trial project');

SELECT TOP 10 *
FROM Client c
JOIN ClientBuilding cb ON c.ClientID = cb.ClientID;

```

Figure 43. ClientBuilding Correction

## 6. Queries using Structured Query Language – SQL

In this section, I will demonstrate, using screenshots and T-SQL, the questions required in Phase 4.

### Query 4. a

Write a query to show details of clients, their property location(s) and the total value of Smart Home designs installed, specifically for those clients who have made the most expensive and the least expensive total values.

```

-- SMART_HOME_LSBU.sql - LAPTOP-PSP6BBV6\SQLEXPRESS.Smart Home LSBU (LAPTOP-PSP6BBV6\raul (68)) - Microsoft SQL Server Management Studio
-- File Edit View Query Project Tools Window Help
-- Smart_Home_LSBU Execute
-- /* 4a. Show details of clients, their property location(s), and the total value of Smart Home designs installed.
-- Limit to clients with the most and least total values. */

WITH Totals AS (
    SELECT
        c.ClientID,
        CONCAT(c.FirstName, ' ', c.LastName) AS ClientName,
        SUM(i.Amount) AS TotalDesignValue
    FROM Client c
    JOIN Invoice i ON c.ClientID = i.ClientID AND i.InvoiceType = 'Design'
    GROUP BY c.ClientID, c.FirstName, c.LastName
),
MinMax AS (
    SELECT
        MIN(TotalDesignValue) AS MinVal,
        MAX(TotalDesignValue) AS MaxVal
    FROM Totals
),
WithAddress AS (
    SELECT
        t.ClientID,
        t.ClientName,
        MIN(b.Address) AS Address,
        t.TotalDesignValue
    FROM Totals t
    LEFT JOIN ClientBuilding cb ON t.ClientID = cb.ClientID
    LEFT JOIN Building b ON cb.BuildingID = b.BuildingID
    GROUP BY t.ClientID, t.ClientName, t.TotalDesignValue
)
SELECT *
FROM WithAddress
WHERE TotalDesignValue = (SELECT MinVal FROM MinMax)
    OR TotalDesignValue = (SELECT MaxVal FROM MinMax)
    ORDER BY TotalDesignValue DESC;
    
```

The screenshot shows the SQL Server Management Studio interface with the query window open. The code is displayed in the center pane, and the results are shown in the bottom pane. The results table has three columns: ClientID, ClientName, and TotalDesignValue. It contains two rows:

	ClientID	ClientName	Address	TotalDesignValue
1	20	Freya Davies	NULL	1100.00
2	8	Isabella King	55 Brixton Hill	750.00

Figure 44. Query 4. A

As seen in Figure 44, the first part of the *ClientDesignTotals* code creates a temporary table that returns the client's ClientID, first name, last name, address, and the total sum of the "Design" invoices. The second part of the code calculates the minimum and maximum *TotalDesignValue* for all clients to identify who spent the most and who spent the least. Finally, the SELECT statement shows us only the clients whose total spending matches the

minimum or maximum value from the entire list. The result shown at the bottom of the code responds to Query 4.a, which displays the client details. The final result indicates that Jake Reed is the client with the highest spending on his Smart Home designs (£1980) and Isabella King is the one who spent the most on her designs (£750).

### Query 4. b

Write a query to calculate individual totals for the number of complete, incomplete and cancelled WiFi product orders from each of its suppliers.

To resolve this query, I had to modify the values in the StockItem table, as it originally lacked an OrderStatus field. As shown in *Figure 45*, the implementation code updates the OrderStatus field in the StockItem table, assigning corresponding values to the Wi-Fi products based on their order status. The OrderStatus values were assigned as 'Complete' for certain items, 'Incomplete' for others, and 'Cancelled' for cancelled orders.

```

Solution1 - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Smart_Home_LSBU Execute
----- ISSUE QUERY 4.B-----
--as i did not have orderstatus i need to aggregate it to StockItemID--
ALTER TABLE StockItem
ADD OrderStatus NVARCHAR(50);

--populating--
UPDATE StockItem SET OrderStatus = 'Complete' WHERE StockItemID IN (1, 3, 6, 10, 12, 15);
UPDATE StockItem SET OrderStatus = 'Incomplete' WHERE StockItemID IN (2, 5, 7, 11, 16, 18);
UPDATE StockItem SET OrderStatus = 'Cancelled' WHERE StockItemID IN (4, 8, 9, 13, 14, 17, 19, 20);

SELECT DISTINCT ItemType FROM StockItem;

--the issue is that i dont have values like router so i have to UPDATE IT--

UPDATE StockItem
SET ItemType = 'WiFi Controller'
WHERE StockItemID IN (1, 2, 3);

UPDATE StockItem
SET ItemType = 'WiFi Repeater'
WHERE StockItemID IN (4, 5, 6);

--QUERY4.B--
SELECT
    s.SupplierID,
    s.SupplierName,
    s.OrderStatus,
    COUNT(*) AS TotalOrders
FROM StockItem s
GROUP BY s.SupplierID, s.SupplierName, s.OrderStatus
ORDER BY TotalOrders DESC;

```

SupplierID	SupplierName	OrderStatus	TotalOrders	
1	Acme Corp	Cancelled	1	
2	Bosch UK	Complete	1	
3	Fibaro	Incomplete	1	
4	1	Libelium	Complete	1
5	6	Philips Hue	Complete	1
6	2	Shenzhen Kingsich	Incomplete	1

*Figure 45. Query 4. B issue*

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists databases, tables, and other objects under the 'Smart\_Home LSBU' database. The central pane displays a corrected query for Figure 46:

```

SELECT DISTINCT ItemType FROM StockItem;

--the issue is that i dont have values like router so i have o UPDATE IT--

UPDATE StockItem
SET ItemType = 'WiFi Controller'
WHERE StockItemID IN (1, 2, 3);

UPDATE StockItem
SET ItemType = 'WiFi Repeater'
WHERE StockItemID IN (4, 5, 6);

--QUERY4.B--
SELECT
    s.SupplierID,
    s.Name AS SupplierName,
    si.OrderStatus,
    COUNT(*) AS TotalOrders
FROM Supplier s
JOIN StockItem si ON s.SupplierID = si.SupplierID
WHERE si.ItemType LIKE '%WiFi%'
GROUP BY s.SupplierID, s.Name, si.OrderStatus
ORDER BY s.Name, si.OrderStatus;

```

The results pane shows the output of the query:

	SupplierID	SupplierName	OrderStatus	TotalOrders
1	1	Aerohive	Cancelled	1
2	3	Bosch UK	Complete	1
3	5	Fibaro	Incomplete	1
4	1	Libelium	Complete	1
5	6	Philips Hue	Complete	1
6	2	Shenzhen Kingtech	Incomplete	1

At the bottom, a message indicates: "Query executed successfully."

Figure 46. Query 4.b corrected

Figure 46 shows the result of query 4.b. The query calculates individual totals for each Wi-Fi product supplier, grouping them by their *OrderStatus* (whether the order is complete, incomplete, or cancelled). This enables the analysis of each supplier's performance and the monitoring of order status based on their current status.

The results show that each supplier has at least one Wi-Fi order, and these are categorized by status. For example, Bosch UK, Libelium, and Philips Hue have full orders, while Fibaro and Shenzhen Kingtech have incomplete orders. This helps LSBU monitor supplier performance and identify those struggling to fulfil orders.

### Query 4. c

Write a query to find the details of specialist staff who are available for booking onto installation jobs this week.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'Smart\_Home LSBU' database is selected. In the center pane, a query window titled 'Diagram.1\*' displays the following T-SQL code:

```

/*
4c - Query to find available specialist staff for installation jobs this week
*/
SELECT
    StaffID,
    CONCAT(FirstName, ' ', LastName) AS StaffName,
    PhoneNumber,
    Email,
    AreaOfExpertise,
    AvailabilityStatus
FROM Staff
WHERE AvailabilityStatus = 'Available'
    AND AreaOfExpertise IS NOT NULL
ORDER BY AreaOfExpertise, LastName;

```

Below the code, a note explains the purpose of the query:

/\*This query returns a list of specialist staff members who are currently marked as available for installation jobs. The result includes their contact details and area of expertise, such as Automation Configuration, Lighting Automation, and IoT Firmware. A total of 11 staff members are shown, which allows LSBU to efficiently schedule the right professionals for upcoming jobs this week.\*/

The results grid shows 11 rows of data:

StaffID	StaffName	PhoneNumber	Email	AreaOfExpertise	AvailabilityStatus
1	Jacob Ross	0784403321	jacob.ross@lsbu-smarthome.co.uk	Automation Configuration	Available
2	Grace James	0752334455	grace.james@lsbu-smarthome.co.uk	Client Training	Available
3	Noah Morris	0734456637	noah.morris@lsbu-smarthome.co.uk	Controller Configuration	Available
4	Mia Young	07422113344	mia.young@lsbu-smarthome.co.uk	Customer Support	Available
5	William Baker	07776543210	william.baker@lsbu-smarthome.co.uk	Data Networking	Available
6	Lily Parker	0747788990	lily.parker@lsbu-smarthome.co.uk	Energy Systems	Available
7	Sophie Murray	07466554433	sophie.murray@lsbu-smarthome.co.uk	IoT Firmware	Available
8	Henry Ward	0759887766	henry.ward@lsbu-smarthome.co.uk	Lighting Automation	Available
9	James Walker	0730011226	james.walker@lsbu-smarthome.co.uk	Sensor Installation	Available
10	Lucas Hall	0745544332	luca.hall@lsbu-smarthome.co.uk	Smart Device Pairing	Available
11	Liam Thompson	0750013987	liam.thompson@lsbu-smarthome.co.uk	System Integration	Available

At the bottom of the results grid, a message indicates: 'Query executed successfully.'

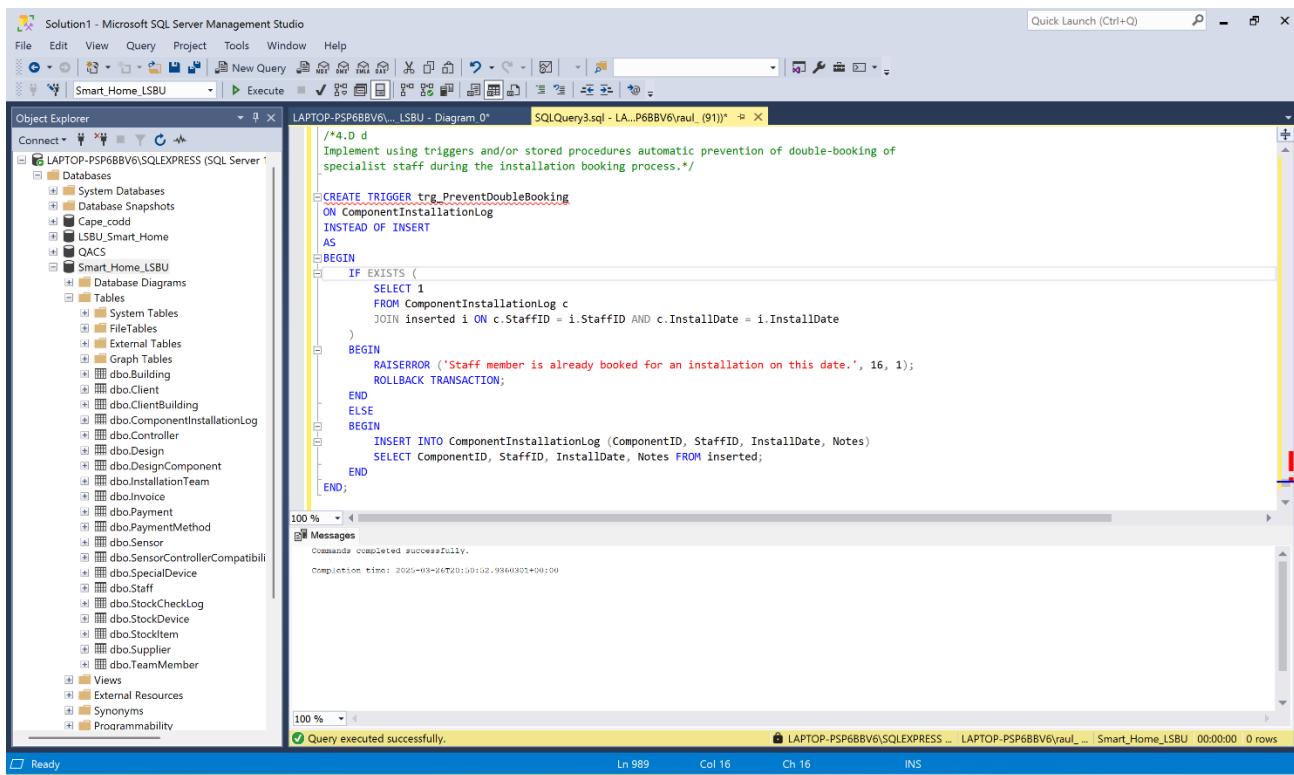
Figure 47. Query 4.c

To resolve query 4.c, as shown in *Figure 47*, the first line of code selects the details of the specialized personnel available this week to be assigned to installation work. The information obtained includes the personnel ID, name, phone number, technical skill, and availability status. This is achieved through a filter with WHERE AvailabilityStatus = 'Available', which ensures that only those staff members who are free to be assigned to installation tasks are returned.

The result identifies 11 available technicians, each with a relevant specialization for Smart Home work, such as sensor automation, controller configuration, smart device support, among others. This allows LSBU to efficiently organize and schedule the appropriate personnel for the installations scheduled for that week.

#### Query 4. d

Implement using triggers and/or stored procedures automatic prevention of double-booking of specialist staff during the installation booking process.



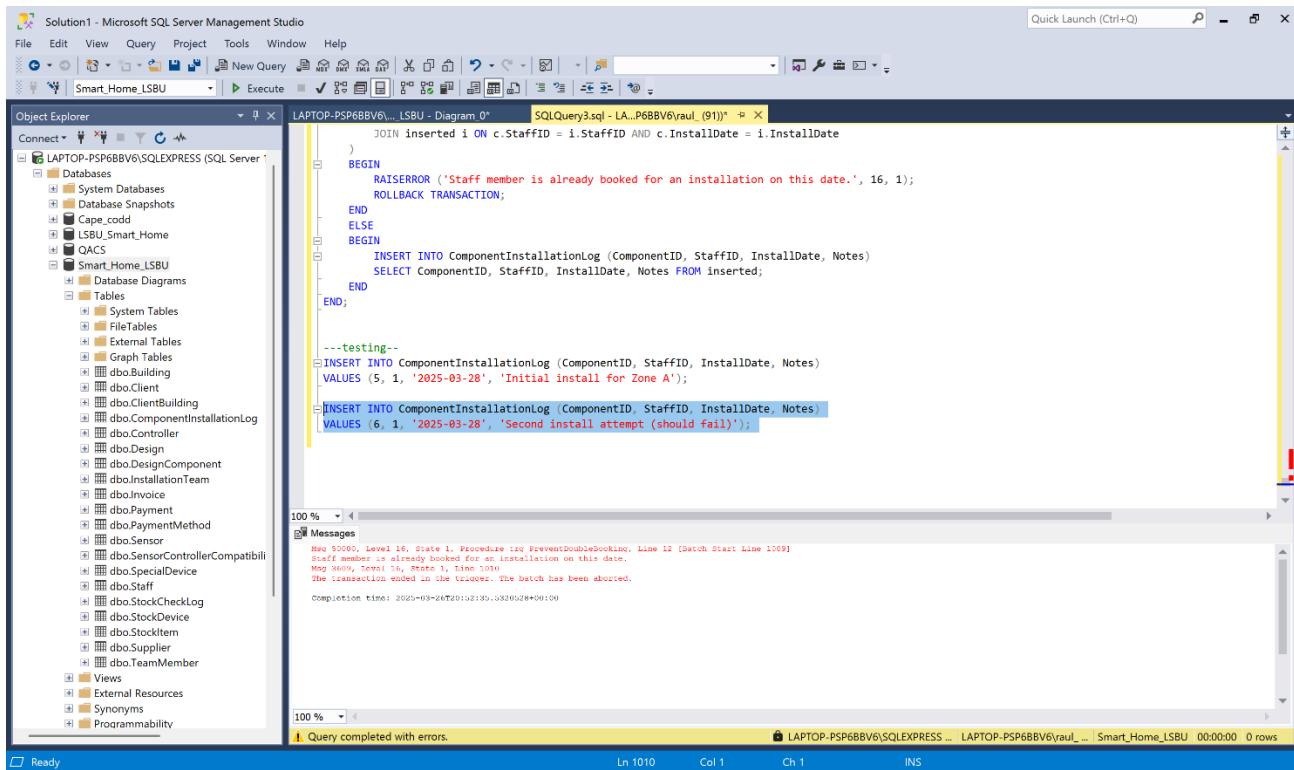
The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. The central pane displays a T-SQL script for creating a trigger named 'trg\_PreventDoubleBooking'. The script uses an INSTEAD OF INSERT trigger on the 'ComponentInstallationLog' table. It checks if the inserted staff member is already booked for the same date. If so, it raises an error and rolls back the transaction. Otherwise, it inserts the new record. The status bar at the bottom indicates the command was executed successfully.

```


/*4.6 d
Implement using triggers and/or stored procedures automatic prevention of double-booking of
specialist staff during the installation booking process.*/
CREATE TRIGGER trg_PreventDoubleBooking
ON ComponentInstallationLog
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM ComponentInstallationLog c
        JOIN inserted i ON c.StaffID = i.StaffID AND c.InstallDate = i.InstallDate
    )
    BEGIN
        RAISERROR ('Staff member is already booked for an installation on this date.', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
        SELECT ComponentID, StaffID, InstallDate, Notes FROM inserted;
    END
END;


```

Figure 48. Trigger Creation



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. The central pane displays a T-SQL script for testing the 'trg\_PreventDoubleBooking' trigger. It attempts to insert two records into the 'ComponentInstallationLog' table. The first insertion succeeds. The second insertion fails because the staff member is already booked for the same date, resulting in an error message and the transaction being rolled back. The status bar at the bottom indicates the command completed with errors.

```


JOIN inserted i ON c.StaffID = i.StaffID AND c.InstallDate = i.InstallDate
)
BEGIN
    RAISERROR ('Staff member is already booked for an installation on this date.', 16, 1);
    ROLLBACK TRANSACTION;
END
ELSE
BEGIN
    INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
    SELECT ComponentID, StaffID, InstallDate, Notes FROM inserted;
END


---testing---
INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
VALUES (5, 1, '2025-03-28', 'Initial install for Zone A');

INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
VALUES (6, 1, '2025-03-28', 'Second install attempt (should fail)');


```

Figure 49. Trigger testing 1

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. The central pane displays a SQL script named 'SQLQuery3.sql' with the following code:

```

LAPTOP-PSP6BBV6\LSBU - Diagram_0*
JOIN inserted i ON c.StaffID = i.StaffID AND c.InstallDate = i.InstallDate
)
BEGIN
    RAISERROR ('Staff member is already booked for an installation on this date.', 16, 1);
    ROLLBACK TRANSACTION;
END
ELSE
BEGIN
    INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
    SELECT ComponentID, StaffID, InstallDate, Notes FROM inserted;
END
END;

---testing---
INSERT INTO ComponentInstallationLog (ComponentID, StaffID, InstallDate, Notes)
VALUES (5, 1, '2025-03-28', 'Initial install for Zone A');

```

The 'Messages' pane at the bottom shows the execution results:

- (1 row affected)
- (1 row affected)
- Completion time: 2025-03-28T11:11:44.4539904+00:00

A green checkmark indicates 'Query executed successfully.'

Figure 50. Trigger testing 2

As shown in *Figure 48*, a trigger called *trg\_PreventDoubleBooking* was implemented to automatically prevent the double assignment of specialized staff to the same team during the installation process. The trigger fires on each insert attempt into the *ComponentInstallationLog* table and checks whether the staff member is already assigned to another team on the same installation date. If a duplicate assignment is detected, an error is thrown, and the transaction is rolled back to ensure there are no scheduling conflicts. Otherwise, the trigger allows the new data to be inserted without issue as shown in *Figure 50* and *Figure 51*.

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery4.sql - LAPTOP-PSP6BBV6\SQLEXPRESS LSBU\_Smart\_Home (LAPTOP-PSP6BBV6\raul\_(52))\* - Microsoft SQL Server Management Studio". The left pane is the Object Explorer, displaying a tree view of database objects including Database Diagrams, Tables, Views, and Security. The right pane is the Query Editor window, titled "LAPTOP-PSP6BBV6\raul\_(52)\*". It contains the following SQL code:

```

--TRYING THE TRIGGER
INSERT INTO TeamMember (TeamID, StaffID, RoleInTeam)
VALUES (1, 1, 'Technician');

INSERT INTO TeamMember (TeamID, StaffID, RoleInTeam)
VALUES (1, 1, 'Installer');

-- It should work if StaffID 2 is not in TeamID 1
INSERT INTO TeamMember (TeamID, StaffID, RoleInTeam)
VALUES (1, 2, 'Installer');

SELECT * FROM TeamMember

```

The status bar at the bottom indicates "Ln 830 Col 1 Ch 1 INS". Below the status bar, a message box says "Query completed with errors." and lists several error messages from the server log:

- Msg 5449, Level 16, State 1, Procedure trg\_PreventDoubleBooking, Line 13 [Batch Start Line 629]
 Staff member already assigned to this team.
- Msg 3609, Level 16, State 1, Line 831
 The transaction ended in the trigger. The batch has been aborted.

The completion time is shown as "Completion time: 2025-03-23T19:41:01.3486054+00:00".

Figure 51. Trying Trigger

## Query 4. e

Write a Stored Procedure that can generate a complete costed device list and total cost for any installation (s) in a specified time related period. The output must also include client and device details. The procedure should be able to accept appropriate parameter values to enable dynamic search by week, month or quarter (3 months) Include appropriate attributes and totals in your report.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. The main pane displays the T-SQL code for creating a stored procedure:

```

CREATE PROCEDURE sp_GetCostedInstallationsByPeriod
    @PeriodType NVARCHAR(10)
AS
BEGIN
    DECLARE @StartDate DATE
    SET @StartDate =
        CASE
            WHEN @PeriodType = 'Week' THEN DATEADD(DAY, -7, GETDATE())
            WHEN @PeriodType = 'month' THEN DATEADD(MONTH, 1, GETDATE())
            WHEN @PeriodType = 'quarter' THEN DATEADD(MONTH, -3, GETDATE())
            ELSE NULL
        END;
    IF @StartDate IS NULL
    BEGIN
        RAISERROR('Invalid period type. Use ''week'', ''month'' or ''quarter''.', 16, 1);
        RETURN;
    END
    SELECT
        c.ClientID,
        CONCAT(c.FirstName, ' ', c.LastName) AS ClientName,
        b.Address,
        dc.ComponentID,
        cl.InstallDate,
        MAX(i.Amount) AS InstallationCost
    FROM Client c
    JOIN ClientBuilding cb ON c.ClientID = cb.ClientID
    JOIN Building b ON cb.BuildingID = b.BuildingID
    JOIN Design d ON b.BuildingID = d.BuildingID
    JOIN DesignComponent dc ON d.DesignID = dc.DesignID
    JOIN ComponentInstallationLog cl ON dc.ComponentID = cl.ComponentID
    JOIN Invoice i ON c.ClientID = i.ClientID AND i.InvoiceType = 'Installation'
    WHERE cl.InstallDate BETWEEN @StartDate AND GETDATE()
    GROUP BY
        c.ClientID,
        CONCAT(c.FirstName, ' ', c.LastName) AS ClientName,
        b.Address,
        dc.ComponentID,
        cl.InstallDate
    ORDER BY cl.InstallDate DESC;
END;

```

The execution results pane at the bottom shows the message "Commands completed successfully." and the completion time "2025-04-02T20:28:29.7303979+01:00".

Figure 52. Procedure Creation

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'Smart\_Home\_LSBU'. The main pane displays the T-SQL code for executing the stored procedure:

```

EXEC sp_GetCostedInstallationsByPeriod @PeriodType = 'month';

```

The execution results pane at the bottom shows the output of the query:

ClientID	ClientName	Address	ComponentID	InstallDate	InstallationCost
1	Emily Watson	45 Victoria Road	5	2025-03-25	1200.00

The message "Query executed successfully." is displayed at the bottom.

Figure 53. Procedure result

The code in *Figure 52* shows a stored procedure called `sp_GetCostedInstallationsByPeriod`, which generates a detailed report listing devices and the total cost of installations within a specified time period. This procedure accepts a `@periodType` parameter, allowing dynamic searches by week, month, or quarter. Depending on the value of this parameter, the start date is adjusted, and the corresponding data is filtered.

Running this procedure returns complete details of the customers, the devices installed, the total installation cost, and the exact dates of installation. This is clearly shown in the results in *Figure 53*.

The code provides a clear and detailed overview of the installations completed during the selected period, making it easier for LSBU to manage the costs associated with Smart Home installations.

## 7. Data Visualization

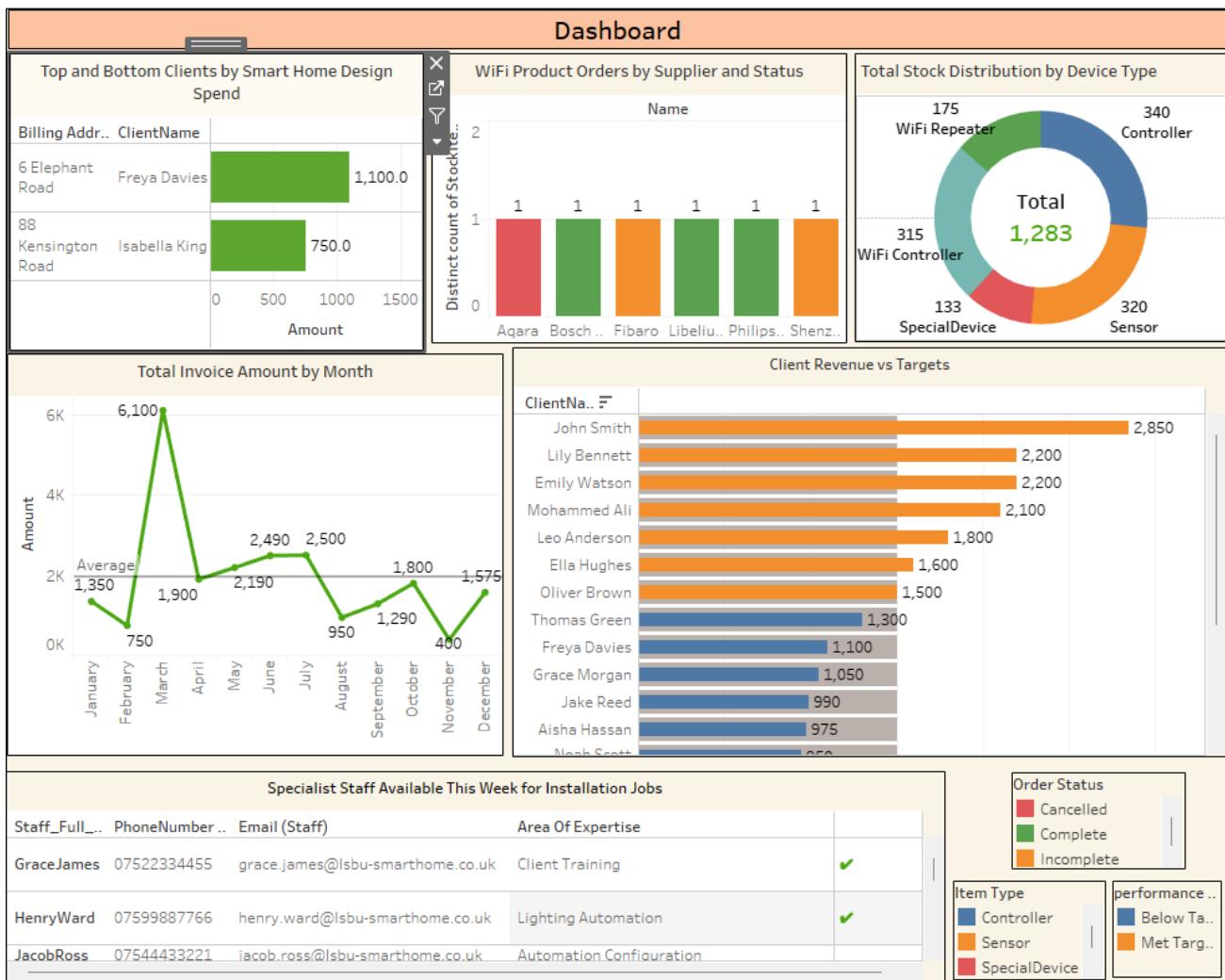


Figure 54. Dashboard

In Figure 54, we can observe the graphical representation of the Dashboard. The first top left graph, named “Top and Bottom clients by Smart Home Design Spend” is the answer for **Query 4. a.** The graph titled “Wi-Fi Product Order by Supplier and Status” represents **Query 4. b.** The graph titled “Specialist Staff Available This Week for Installation Jobs” represents **Query 4.c.**

The next graphs were created randomly by me. The graph titled “Total stock distribution by Device Type” represents the sum of the quantity available for each device, illustrating how stock is distributed. This graph aims to provide information on the current inventory levels of various smart home devices.

The following graph, titled “Total Invoice Amount by Month,” illustrates the monthly revenue generated. The line chart tracks total invoice amounts every month, highlighting revenue trends throughout the year. Furthermore, the dotted trendline might show average performance.

The last graph with name “Client Revenue vs Targets” compares the actual revenue per client against a predefined revenue target (1500). Each client’s actual revenue (coloured bar) is compared to a standard target (background bar). Colour indicates whether they met or missed the target. Useful for performance tracking and client segmentation.

The dashboard can be accessed on the following GitHub repository:

<https://raulcimpe.github.io/Dashboard/>

## 8. Video Demonstration

Microsoft:

<https://stulsbuac-my.sharepoint.com/:v/>

Google Drive:

<https://drive.google>

## 9. References

Dybka, P. (2016). *Crow's Foot Notation*. [online] Vertabelo Data Modeler. Available at: <https://vertabelo.com/blog/crow-s-foot-notation/>.

Visual Paradigm (2019). *What is Entity Relationship Diagram (ERD)?* [online] Visual-paradigm.com. Available at: <https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/>.

Staff, F. (2024). *Understanding Functional Dependencies in Software Engineering*. [online] As We May Think — products & tools for thought. Available at: <https://fibery.io/blog/product-management/functional-dependencies-explained/>.

Microsoft (2024). *Database normalization description*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description>.

## 10. Appendix

### Originality & Use of Generative AI Statement

I understand that using the work or ideas of others, including AI-generated content, without proper acknowledgement is considered academic misconduct.

I confirm that this submission is my own original work, and that all sources and references have been included in line with LSBU Academic Regulations.

### DECLARATION OF AI USE:

I **DID** use Generative AI technology in the development, writing, or editing of this assignment.

#### 1. Tools used and purpose

I used ChatGPT mainly for idea generation, technical clarification, and general writing support. It helped me brainstorm database structure, refine explanations of certain concepts (like normalization and functional dependencies), and polish grammar in a few parts.

Additionally, Grammarly was used for checking English grammar and improving sentence structure, especially when translating from Spanish to English since English is not my first language.

#### 2. Sections supported by AI

AI tools supported parts of:

- Clarifying technical concepts (ERD, functional dependencies)
- Some of the SQL query explanations
- Minor parts of the writing and formatting

All technical implementation, data modelling, and database creation was done manually by me.

#### 3. Example prompts used

- “Explain how a trigger can prevent staff double booking.”
- “How can I describe a normalized ERD in simple academic language?”
- “Translate this sentence from Spanish to English while keeping technical accuracy”

#### 4. Reflection on using AI

Using AI helped speed up the writing process and provided quick support when I needed to check grammar or describe technical parts more clearly. While helpful, I still made sure to review and edit everything based on my own understanding of the case study and project. It was a tool to support my work.

By including this statement in my coursework submission, I attest that the information provided in this Originality and Use of Generative AI Statement is accurate and complete to the best of my knowledge. I understand that providing false information is a violation of the LSBU Academic Regulations and may result in academic and/or disciplinary consequences.