Name: Hinda Nguyen

Date: September 10th, 2025

Course: IT FDN 130

The Final Paper

Contents

*Note: This paper includes short reflections for weekly topics, mainly for my own review. A more comprehensive and concise version will be shared on the “Lessons Learned” discussion board. The shortened version is essentially the “Final Reflection and Next Steps” section.

Introduction

When I began this course, I had only a basic understanding of SQL and viewed it primarily as a tool for retrieving data. I quickly realized it was much broader, encompassing database design, keys, normalization, and advanced topics such as joins, subqueries, functions, and stored procedures. At first, the pace of the course felt challenging, but the step-by-step progression gradually built my confidence.

The hands-on assignments were especially valuable. Practicing CRUD operations, experimenting with joins, and writing functions showed me how theory translates into practical problem-solving. Tools like Tableau and GitHub added new perspectives, connecting database work to visualization and collaboration.

This paper reflects on that journey, from early uncertainty to growing confidence, and how these skills will support me in future academic and professional work.

Week 1: Creating Databases and Tables

Week one introduced the fundamentals of data management, focusing on tables and databases. We also discussed normalization, which reduces duplication and improves integrity by structuring data into well-organized tables, typically up to Third Normal Form (3NF). Together, these concepts form the foundation for reliable and consistent data management in modern systems.

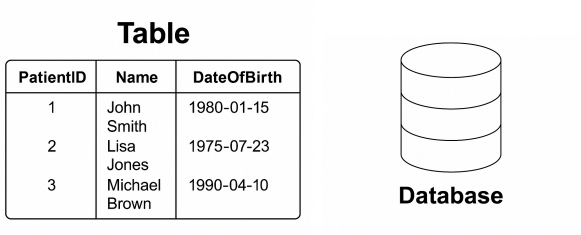
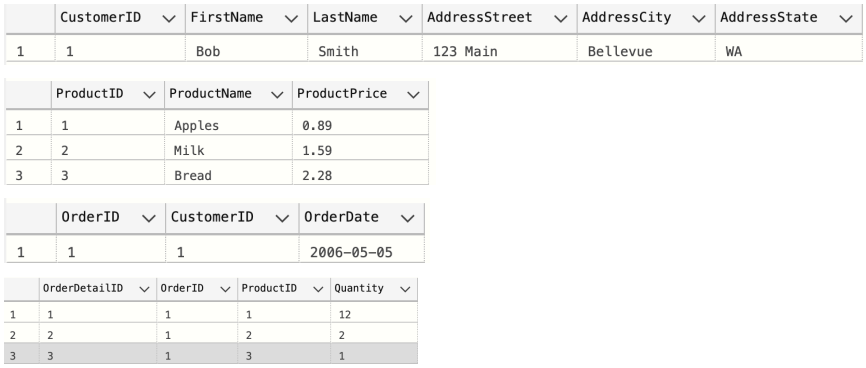


Figure 1. ***Example of a table in a relational database (left) and a symbolic representation of a database (right).***



	CustomerID	FirstName	LastName	AddressStreet	AddressCity	AddressState
1	1	Bob	Smith	123 Main	Bellevue	WA

	ProductID	ProductName	ProductPrice
1	1	Apples	0.89
2	2	Milk	1.59
3	3	Bread	2.28

	OrderID	CustomerID	OrderDate
1	1	1	2006-05-05

	OrderDetailID	OrderID	ProductID	Quantity
1	1	1	1	12
2	2	1	2	2
3	3	1	3	1

Figure 2. Example of a ***normalized table structure.***

At first, these ideas felt abstract, especially normalization. Breaking one large table into several smaller ones seemed counterintuitive, and I struggled to see why it was necessary. However, working with ERDs and examples helped me understand how normalization reduces redundancy and prevents errors. Once I saw how the pieces connected, I began to appreciate the logic behind relational databases and felt more confident about building a solid foundation for later topics.

Week 2: Common DB Options

Week two expanded on the ideas in week one by introducing database constraints: rules that enforce accuracy, consistency, and integrity within relational designs. Constraints such as NOT NULL, CHECK, UNIQUE, PRIMARY KEY, and FOREIGN KEY help further prevent errors, maintain referential integrity, and reduce reliance on application-level validation. This week also introduced abstraction layers through views, which simplify querying, enhance security, and provide consistent access to complex data structures.

Constraint Type	Purpose
NOT NULL	Ensures a column must always have a value (no missing data).
CHECK	Limits values in a column to those that meet a specific condition (e.g., price must be > 0).
UNIQUE	Prevents duplicate entries in a column or combination of columns.
PRIMARY KEY	Uniquely identifies each row in a table and ensures no duplicates or NULLs.
FOREIGN KEY	Enforces relationships between tables by ensuring values match a primary/candidate key in another table.

Table 1. ***Common Constraint Types***

Furthermore, planning tools such as Entity-Relationship Diagrams (ERDs) and Meta-Data Worksheets guided the design process by visualizing table relationships and providing detailed column specifications. My favorite part of this week was creating ERDs. It felt almost like designing a blueprint. I enjoyed seeing how entities connected and how design choices shaped the database structure. For me, this made the abstract rules of normalization and constraints feel much more practical and even fun.

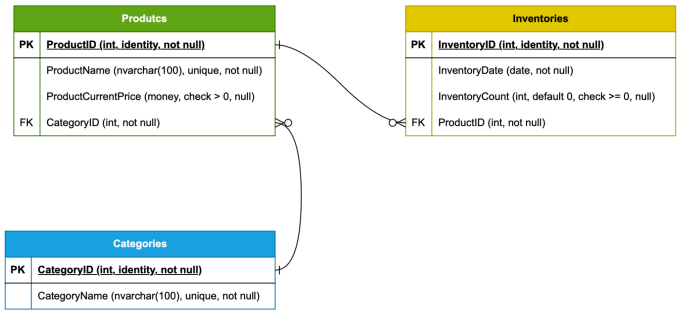


Figure 3. ***Example of an ERD diagram.***

	A	B	C	D	E	F	G	H
1	**SourceObjectName**	**IsNullable**	**DataType**	**Primary Key**	**Unique**	**Foreign Key**	**Check**	**Default**
2	Categories.CategoryID	No	int	**Yes**	No	No	No	**Yes (identity)**
3	Categories.CategoryName	No	nvarchar(100)	No	Yes	No	No	No
4	Products.ProductID	No	int	**Yes**	No	No	No	**Yes (identity)**
5	Products.ProductName	No	nvarchar(100)	No	**Yes**	No	No	No
6	Products.ProductCurrentPrice	**Yes**	money	No	No	No	**Yes (> 0)**	No
7	Products.CategoryID	No	int	No	No	**Yes**	No	No
8	Inventories.InventoryID	No	int	**Yes**	No	No	No	**Yes (identity)**
9	Inventories.InventoryDate	No	date	No	No	No	No	No
10	Inventories.InventoryCount	**Yes**	int	No	No	No	**Yes (>= 0)**	**Yes (0)**
11	Inventories.ProductID	No	int	No	No	**Yes**	No	No

Figure 4. ***Example of a Meta-Data worksheet.***

Week 3: Selecting Data

This week focused on retrieving data with the SELECT statement. We established how clauses such as FROM, WHERE, and ORDER BY work together to filter and organize results. Remembering the order of clauses felt tricky, but practicing different queries helped me see how flexible SQL can be. I enjoyed experimenting with conditions and sorting because it gave me a sense of control over the data, and made querying feel more intuitive.

```
SELECT p.ProductName, AVG(i.InventoryCount) AS [AvgInventoryCount]
    FROM vInventories i
    JOIN vProducts p on i.ProductID = p.ProductID
    WHERE p.CategoryID = (
        SELECT CategoryID
            FROM vCategories
            WHERE CategoryName = 'Seafood'
    )
    AND (MONTH(i.InventoryDate) = 1 OR MONTH(i.InventoryDate) = 2)
    GROUP BY p.ProductName
    ORDER BY p.ProductName;
go
```

Figure 5. ***Example usage of the SELECT clause.***

Week 4: Processing Data

This week introduced the four core SQL statements—INSERT, SELECT, UPDATE, and DELETE—which make up the CRUD model. We also learned about transactions, which group multiple statements together so they either all succeed or roll back errors. Transactions actually felt the most familiar to me since they reminded me of using loops in Python, where multiple steps are handled as a unit. That connection made it easier to understand how they protect data integrity. Learning about identity features, such as auto-incrementing keys, also helped me see how databases manage relationships more efficiently.

```
BEGIN TRY
   BEGIN TRAN
      INSERT INTO Products (...) VALUES (...);
      INSERT INTO Inventories (...) VALUES (...);
   COMMIT
END TRY
BEGIN CATCH
   ROLLBACK
   PRINT ERROR_MESSAGE();
END CATCH
```

Figure 6. Example *template for a transaction.*

Week 5: Joins and Subqueries

The fifth week focused on JOINs and subqueries for combining data across tables. INNER, OUTER, and CROSS JOINs each served different purposes, while self-joins showed how a table can reference itself. Initially, the variety of JOIN types felt overwhelming, but practicing them helped me see how powerful they are for building meaningful results from distributed data. The diagrams below were especially helpful. I realized that joins are what truly make relational databases "relational."

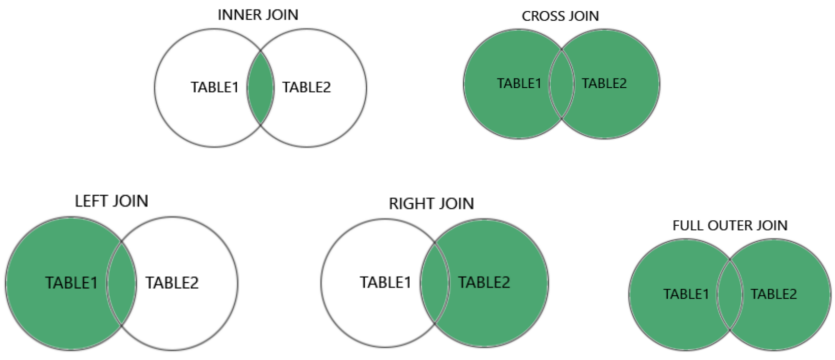


Figure 7. ***Diagrams of different types of joins.***

Week 6: Views

Building on joins and subqueries, this week introduced views, functions, and stored procedures as tools to simplify and centralize SQL logic. Views acted as virtual tables that streamlined complex queries and, more importantly, protected sensitive information. Functions provided reusable calculations or filtered results, while stored procedures combined multiple SQL statements with logic and transactions to handle larger operations efficiently. I found this week fairly easy since the coding felt similar to what we had already been practicing, but I also realized how commanding views were in shaping and securing data access. This balance of familiarity and new capability made the material engaging and rewarding.

```sql
CREATE VIEW dbo.vInventoriesByProductsByCategoriesByEmployees
WITH SCHEMABINDING
AS
SELECT
    c.CategoryID,
    c.CategoryName,
    p.ProductID,
    p.ProductName,
    i.InventoryID,
    i.InventoryDate,
    i.[Count],
    e.EmployeeID,
    e.EmployeeFirstName,
    e.EmployeeLastName,
    -- Employee full name
    EmployeeName = e.EmployeeFirstName + ' ' + e.EmployeeLastName,
    -- Manager full name
    ManagerName = m.EmployeeFirstName + ' ' + m.EmployeeLastName
FROM dbo.vCategories c
JOIN dbo.vProducts p ON c.CategoryID = p.CategoryID
JOIN dbo.vInventories i ON p.ProductID = i.ProductID
JOIN dbo.vEmployees e ON i.EmployeeID = e.EmployeeID
LEFT JOIN dbo.vEmployees m ON e.ManagerID = m.EmployeeID;
GO

SELECT *
FROM dbo.vInventoriesByProductsByCategoriesByEmployees
ORDER BY CategoryName, ProductName, InventoryID, EmployeeName;
```

Figure 8. ***A view can join multiple base views or tables to centralize query logic.***

Week 7: Functions

This week introduced User-Defined Functions (UDFs), which package reusable logic for calculations, formatting, or filtering. We worked with three main types: scalar functions that return a single value, inline table-valued functions for efficient result sets, and multi-statement table-valued functions for more complex operations similar to the KPI function we built. I appreciated how UDFs made queries cleaner and more consistent across different tasks. Compared to earlier weeks, this felt like a natural extension of what we had already learned, but it also showed how SQL can scale with complexity in real-world applications.

```sql
CREATE OR ALTER FUNCTION fProductInventoriesWithPreviousMonthCountsWithKPIs
(
    @KPIFilter INT = NULL
)
RETURNS TABLE
AS
RETURN
(
    SELECT
        ProductName,
        InventoryDate,
        InventoryCount,
        PreviousMonthCount,
        CountVsPreviousCountKPI
    FROM vProductInventoriesWithPreviousMonthCountsWithKPIs
    WHERE @KPIFilter IS NULL OR CountVsPreviousCountKPI = @KPIFilter
);

go
```

Figure 9. ***Example of a user-defined function.***

Week 8 & 9: Stored Procedures

These weeks focused on stored procedures and transactional processing. Stored procedures enabled me to combine queries, logic, and error handling into reusable, modular code, while transactions ensured operations succeeded or rolled back together to maintain data integrity. We also began our final project, and I felt confident rather than nervous because I was well-prepared. The workload was a bit intimidating, but having three weeks to complete it felt manageable, and I trusted my abilities to complete everything.

```sql
-- Create a stored procedure to insert a new product
CREATE PROCEDURE pAddProduct
   @ProductName NVARCHAR(50),
   @UnitPrice DECIMAL(10,2),
   @CategoryID INT,
   @NewProductID INT OUTPUT  -- Output parameter to return the new ProductID
AS
BEGIN
   SET NOCOUNT ON;

   BEGIN TRY
      -- Start a transaction to ensure atomicity
      BEGIN TRANSACTION;

      -- Insert the new product into the Products table
      INSERT INTO Products (ProductName, UnitPrice, CategoryID)
      VALUES (@ProductName, @UnitPrice, @CategoryID);

      -- Capture the ID of the newly inserted product
      SET @NewProductID = SCOPE_IDENTITY();

      -- Commit the transaction
      COMMIT TRANSACTION;
   END TRY
   BEGIN CATCH
      -- Roll back if an error occurs
      ROLLBACK TRANSACTION;
```

```
        PRINT 'Error: ' + ERROR_MESSAGE();
    END CATCH
END;
GO

DECLARE @NewID INT;

-- Call the procedure and pass the output parameter
EXEC pAddProduct
    @ProductName = 'Green Tea',
    @UnitPrice = 12.99,
    @CategoryID = 3,
    @NewProductID = @NewID OUTPUT;

-- Check the returned ProductID
PRINT 'The new ProductID is: ' + CAST(@NewID AS NVARCHAR(10));
```

Figure 10. ***Example of a stored procedure and how to execute it.***

Week 10: Database Applications

In the final week, we focused on applying SQL skills through business intelligence tools like Tableau. Connecting directly to databases, Tableau allowed me to visualize data and build dashboards, turning raw numbers into actionable insights. I enjoyed seeing how concepts like KPIs and performance metrics come to life visually. This week helped me understand the real-world impact of databases and reinforced how the skills I've been learning can support data-driven decision-making. Nevertheless, I will definitely be needing more practice with visualization.

Final Reflection and Next Steps

Over the quarter, the course provided a strong foundation in relational database concepts, SQL programming, and data management best practices. It began with the basics—SELECT statements, CRUD operations, and joins—and advanced to topics such as user-defined functions, stored procedures, and transaction management. The progression was structured and subsidized by hands-on assignments that reinforced each concept and demonstrated its application in real-world database work.

What stood out to me was learning how to make code more efficient and reusable through views, functions, and stored procedures. These tools not only streamlined my queries but also gave me insight into how professional developers maintain and secure their databases. Working with Tableau was another highlight. It showed me how data can be transformed from raw numbers into meaningful, visual confirmation that supports decision-making. Using GitHub throughout the course also taught me the importance of version control and collaboration, skills that I know will carry over into any technical role.

While my previous experience with coding languages such as Python, C++, and Java helped me grasp SQL concepts more quickly, especially logic and control structures, the coursework and lectures were thorough and structured enough that someone with no prior coding experience could still follow along and succeed. My only regret is not being able to attend the live Zoom sessions due to work-time conflicts, but the recordings were great.

Looking ahead, I'm excited to apply these skills in both academic and professional projects. I want to design databases that are not only efficient, but also intuitive and user-friendly. I also plan to keep building on what I learned with Tableau by exploring more advanced business intelligence tools and data visualization techniques. Staying current with best practices in database security, performance tuning, and cloud-based solutions will also be essential to maintaining relevance in a rapidly evolving field. Overall, this course has given me both the technical foundation and the confidence to continue growing as someone who can transform data into practical, actionable knowledge.