

# TAMING THE DATA TIDES:

Streamlining Database Workflows  
with Stored Procedures and GenAI

## **Presenters:**

Farazuddin Mohammed – Graduate Research Assistant, PAIRS, TAMUCC

Jiashi Zhao, EdD – Lead Data Analyst, PAIRS, TAMUCC

# Streamlining Database Operations

## Train with GenAI

Upskilling team using AI tools for advanced queries.

## Achieve Impact

Realizing faster operations and reduced errors.

## Implement Automation

Transforming workflows with Stored Procedures and Dynamic SQL.

## Identify Challenges

Recognizing inefficiencies in manual database tasks.



# AGENDA

- BACKGROUND
  - TEXAS A&M UNIVERSITY-CORPUS CHRISTI
  - PAIRS DEPARTMENT
- CHALLENGES IN DATABASE OPERATIONS
- AUTOMATION WITH STORED PROCEDURES AND DYNAMIC SQL
- IMPACT ON WORKFLOW EFFICIENCY
- LEVERAGING GENAI TOOLS FOR SQL CRAFTING
- Q&A SESSION



# TEXAS A&M UNIVERSITY – CORPUS CHRISTI

- "The Island University"
- Established in 1947
- Research II Institution
- Hispanic Serving Institution (HSI)
- 6 colleges and 1 school
- Fall 2024 Enrollment: 11,266
- Offer bachelor's, master's, and doctoral degrees

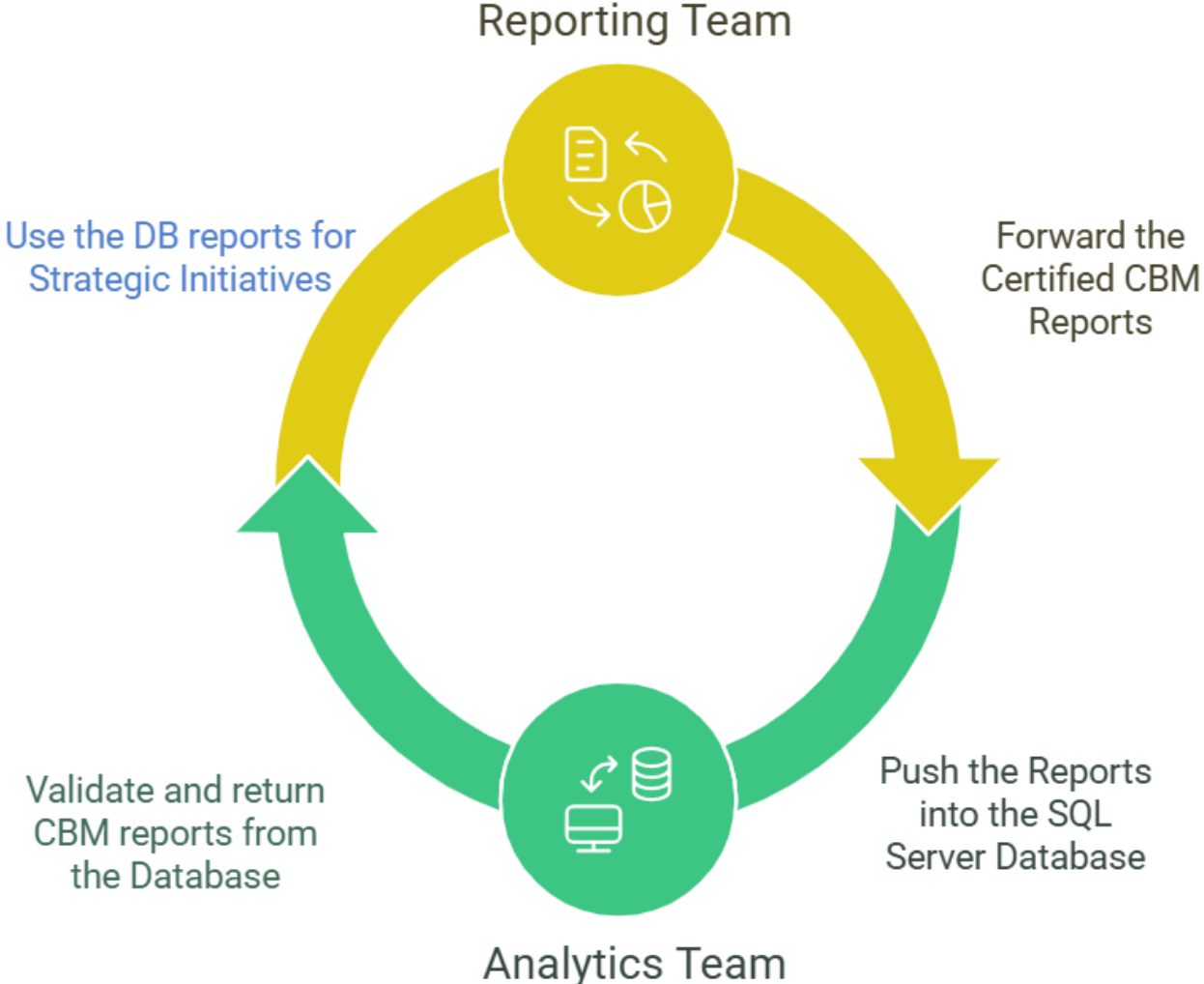


# PAIRS –TAMUCC (Planning, Analytics, Institutional Research, & Strategic Initiatives)

- THREE DIFFERENT AREAS
  - PLANNING AND INSTITUTIONAL RESEARCH
  - ANALYTICS
  - STRATEGIC INITIATIVES

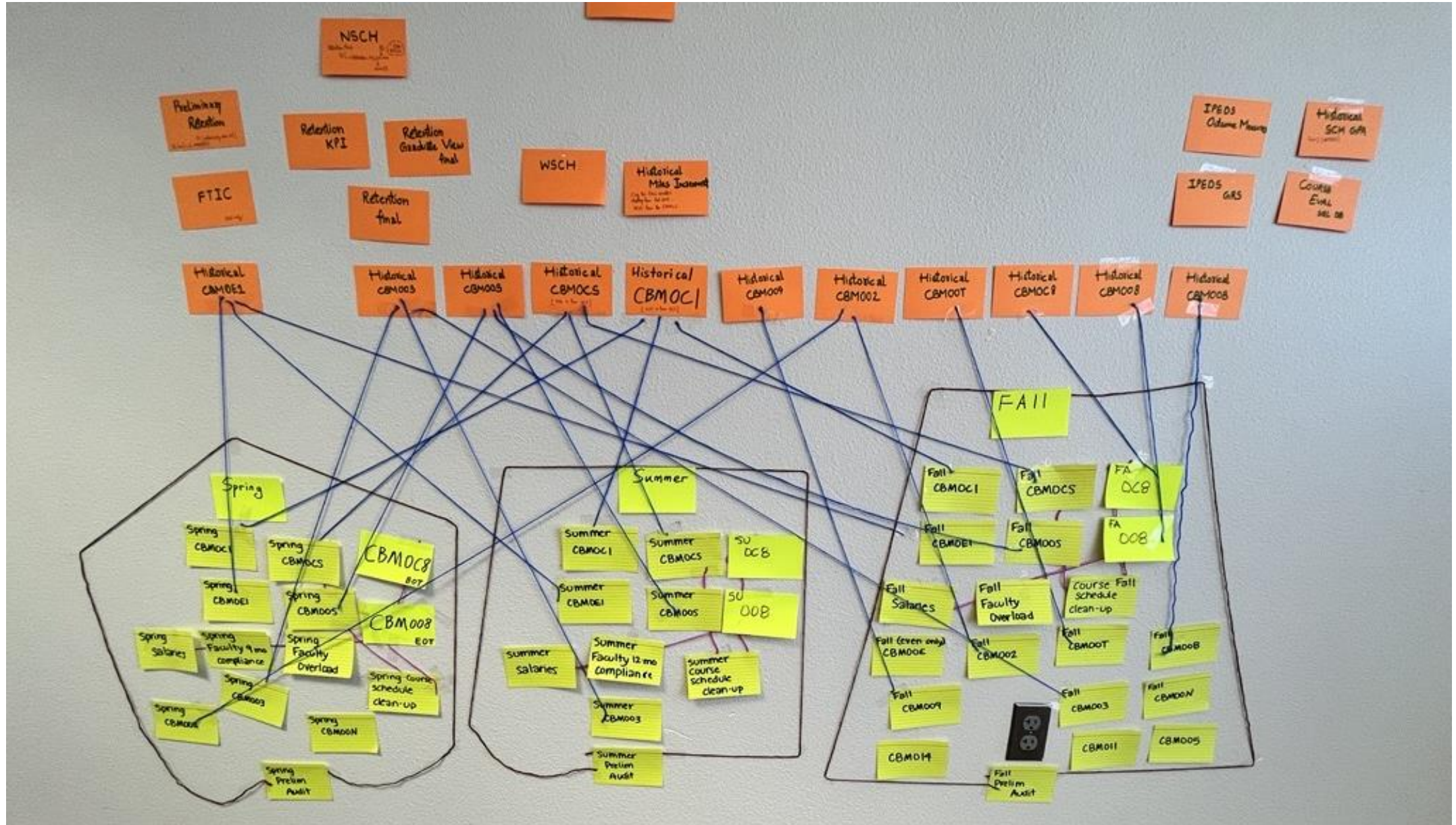


# CBM REPORTS FLOW



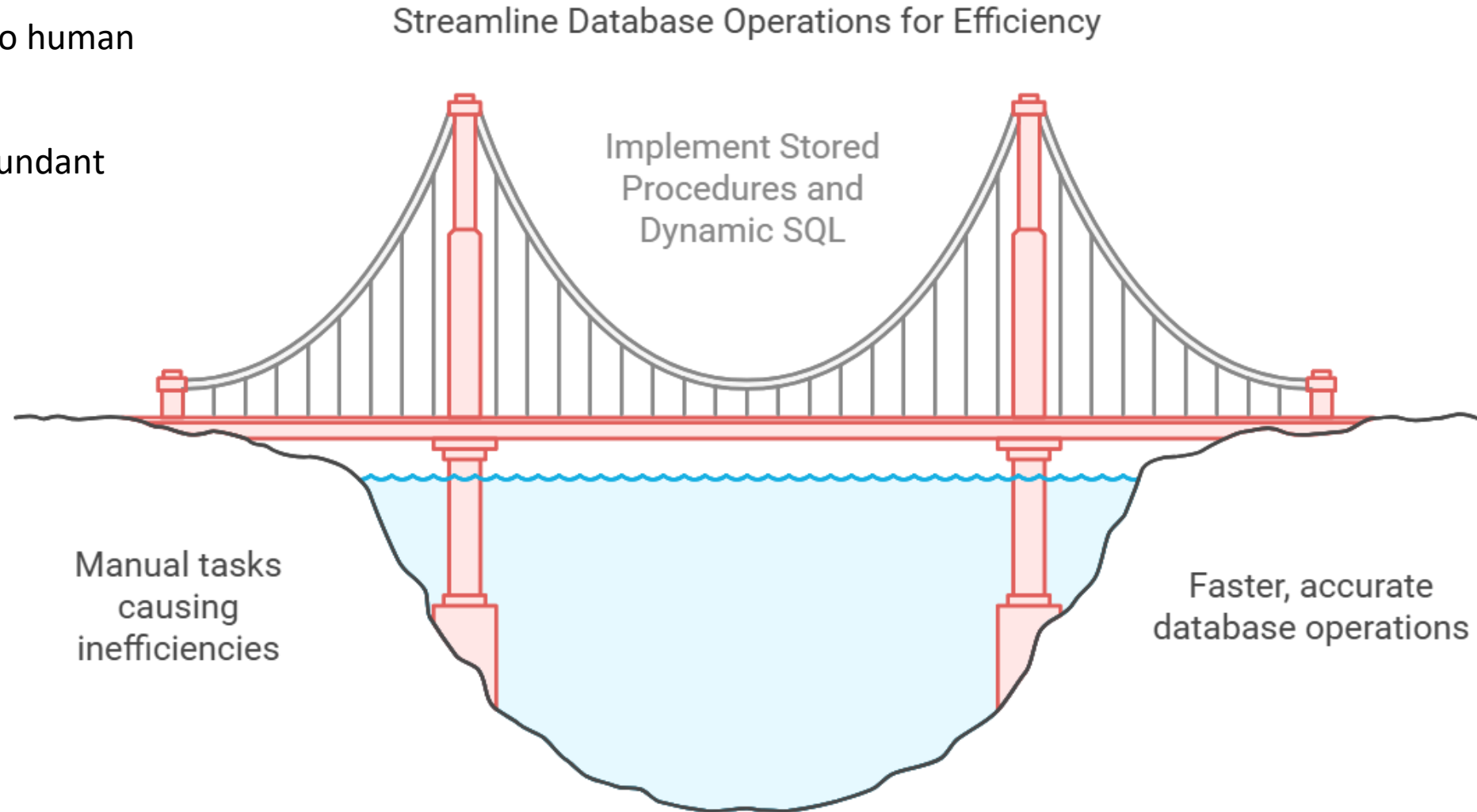


# A CLOSER LOOK

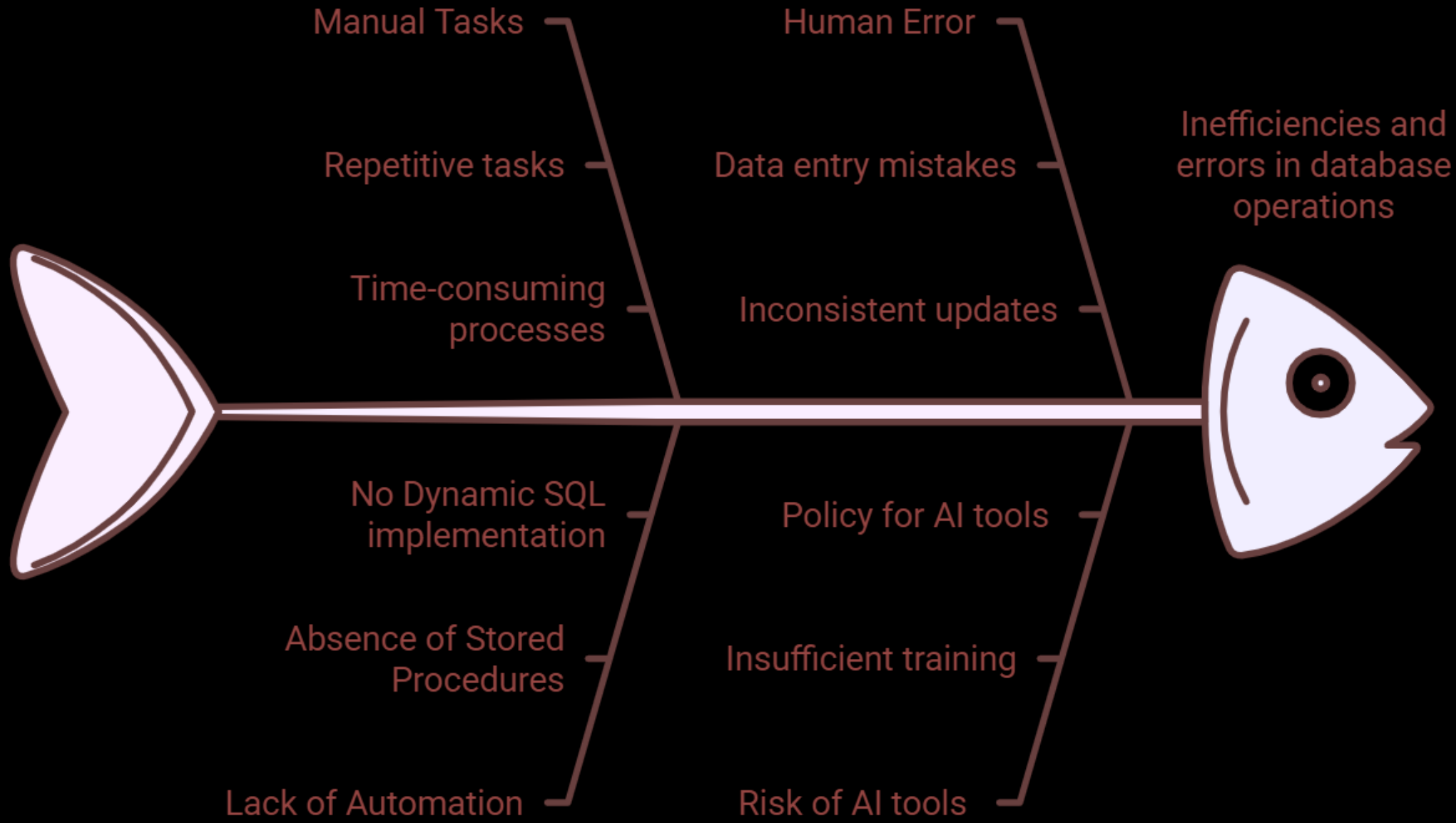


# CHALLENGES IN DATABASE OPERATIONS

- Time-consuming recurring tasks (e.g., data appends, updates, validations)
- Manual SQL queries prone to human error
- Low productivity due to redundant workflows





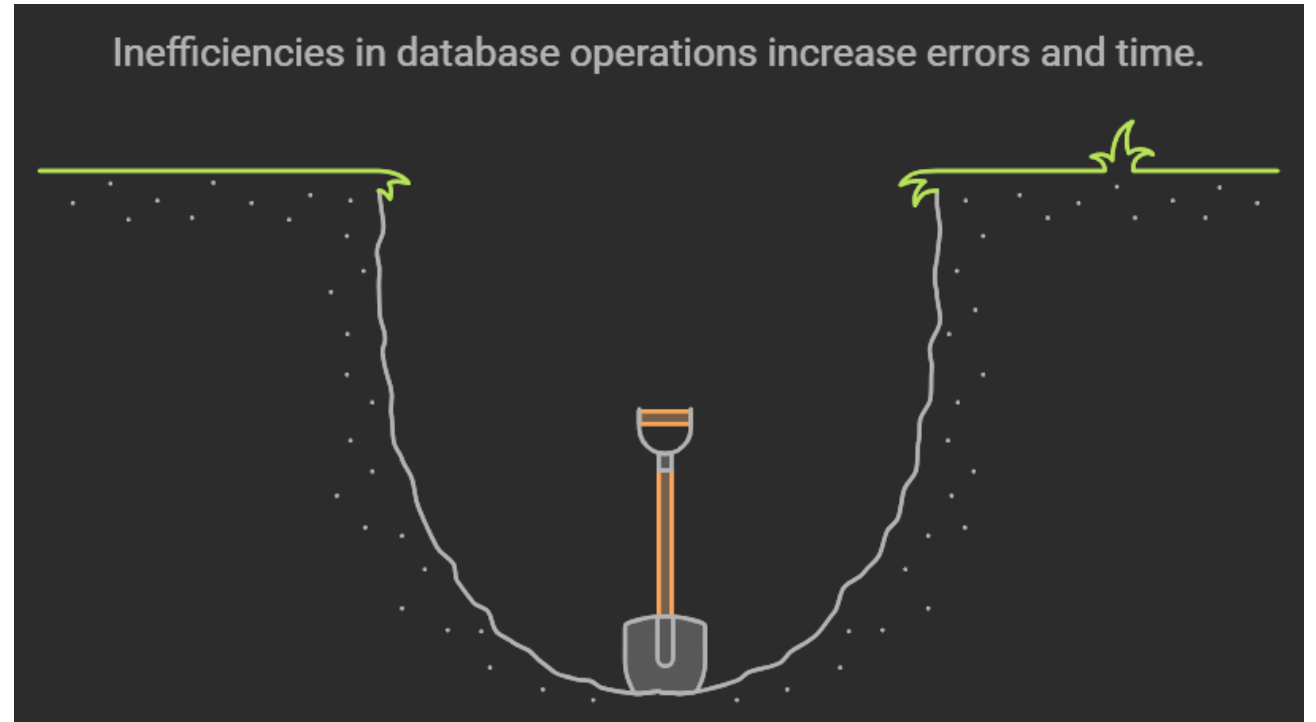


# OUR OBJECTIVES

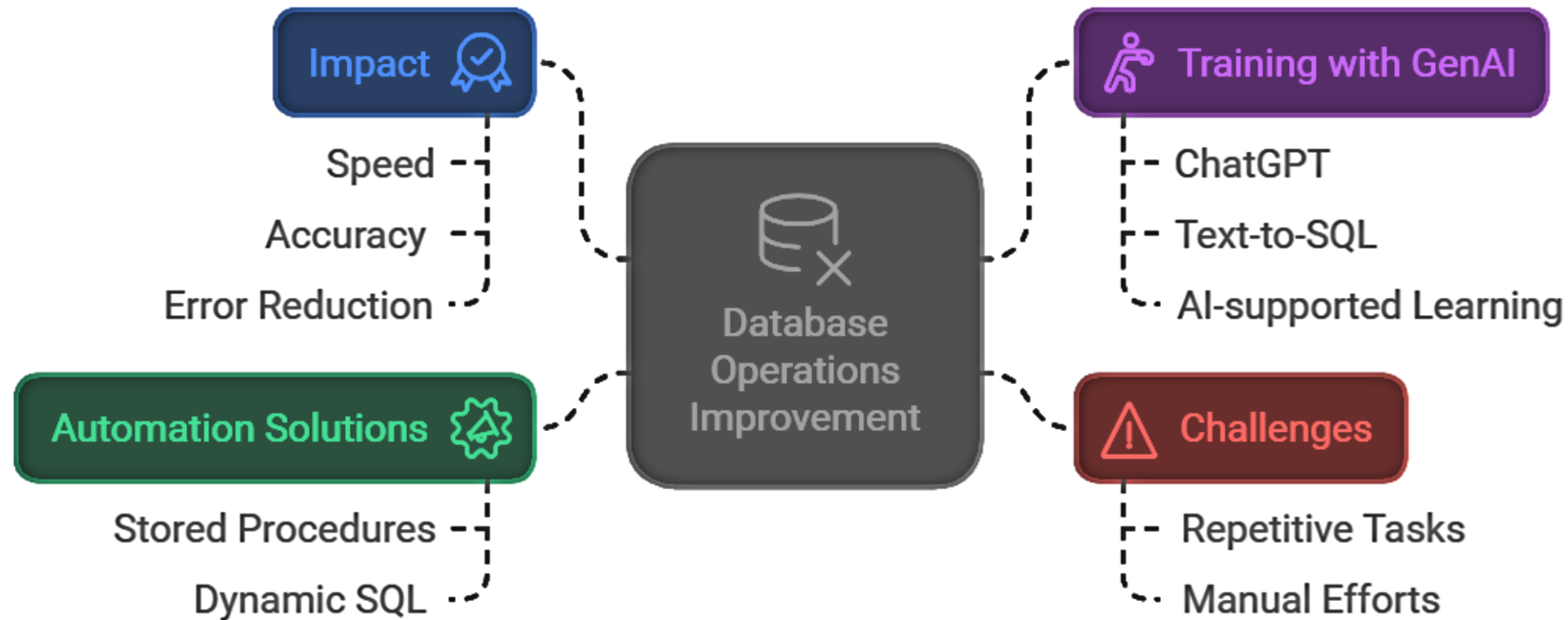
Increasing need for efficient database management.

Key objective:

- Automate repetitive tasks
- Reduce errors
- Save time



# DATABASE OPERATIONS IMPROVEMENT



- We addressed repetitive tasks by implementing Stored Procedures and Dynamic SQL to automate processes, significantly improving speed, accuracy, and minimizing errors.
- Additionally, we utilized GenAI tools like ChatGPT to enhance our SQL skills through AI-supported learning and efficient Text-to-SQL query generation.

# AUTOMATION SOLUTIONS

- **Stored Procedures:**

Predefined SQL code to automate repetitive tasks and handle multiple inputs efficiently.

- Automating repetitive tasks
- Parameterizing queries to handle various inputs.

- **Dynamic SQL:**

SQL statements built and executed at runtime, offering flexibility to manage complex queries across multiple datasets.

- Flexibility to build and execute complex queries dynamically
- Handling multiple datasets efficiently

## IMPLEMENTATION OF STORED PROCEDURES & DYNAMIC SQL

- We implemented **Stored Procedures** and **Dynamic SQL** to streamline table append operations, optimize updates, and enhance query validation across multiple datasets.

Table/Database	Usage
CBM0C1	Automated 0c1 data updates using Dynamic SQL and Stored Procedures.
cbmS	Automated cbmS data updates using Dynamic SQL and Stored Procedures.
Retention_final (New Term Data Updates after Append)	Implemented using Dynamic SQL, Stored Procedures for new term updates.
Retention_final (Cohort_Type Column)	Used multiple queries encapsulated in Stored Procedures
Retention_KPI	Relies exclusively on <b>Dynamic SQL</b> for flexibility and reduced manual queries.

- Dynamic SQL Usage:
- Across Validations and Debugging Queries:
  - Applied throughout the Retention database to handle complex, multi-term data (Term 1 to 24)
- Benefits:
  - Improved writability and readability.
  - Ensures no edge cases are missed.
  - Faster, automated validation compared to manual checks used previously

# IMPACT AND RESULTS

- **Faster data updates and validations.**
- **Significant reduction in completion time for key processes.**
- **Enhanced productivity of database administrators.**
- **Fewer human errors with automated workflows**



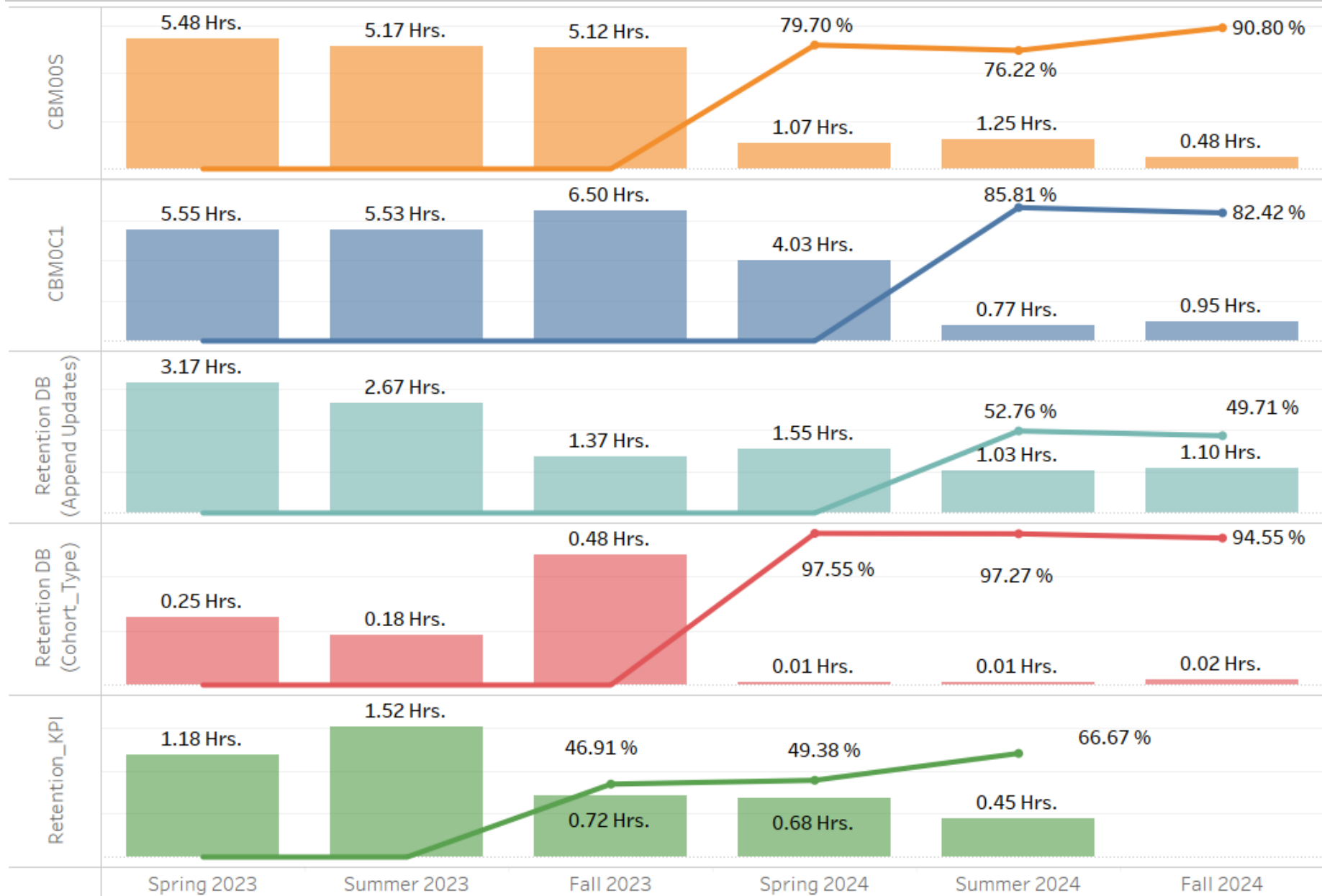
Simple  
SQL Queries  
for DB Updates!



Stored  
Procs &  
Dynamic SQL  
for DB Updates!

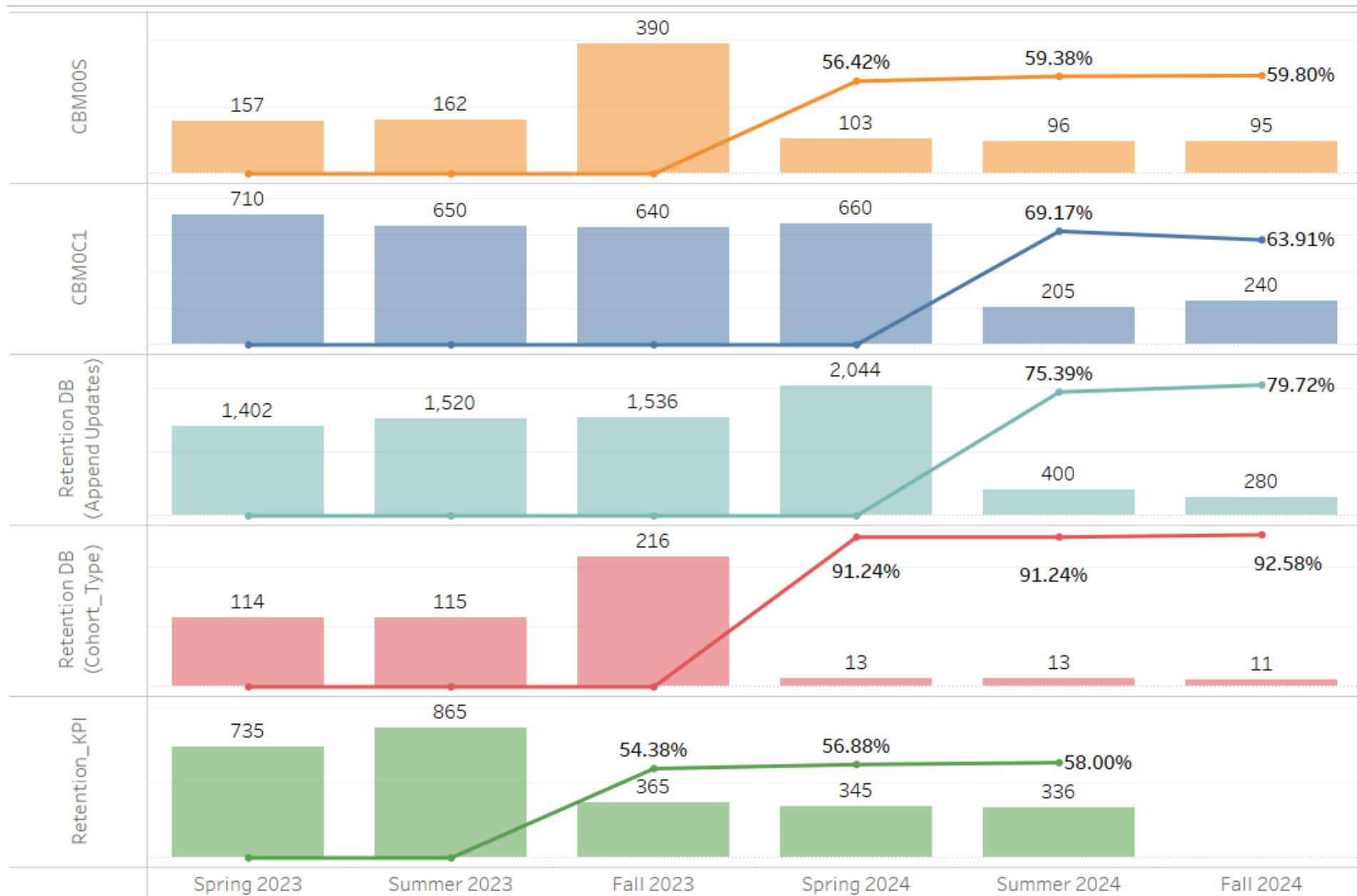
# IMPACT AND RESULTS

## Stored Procs & Dynamic SQL Impact on DB Run Time Gains



# IMPACT AND RESULTS

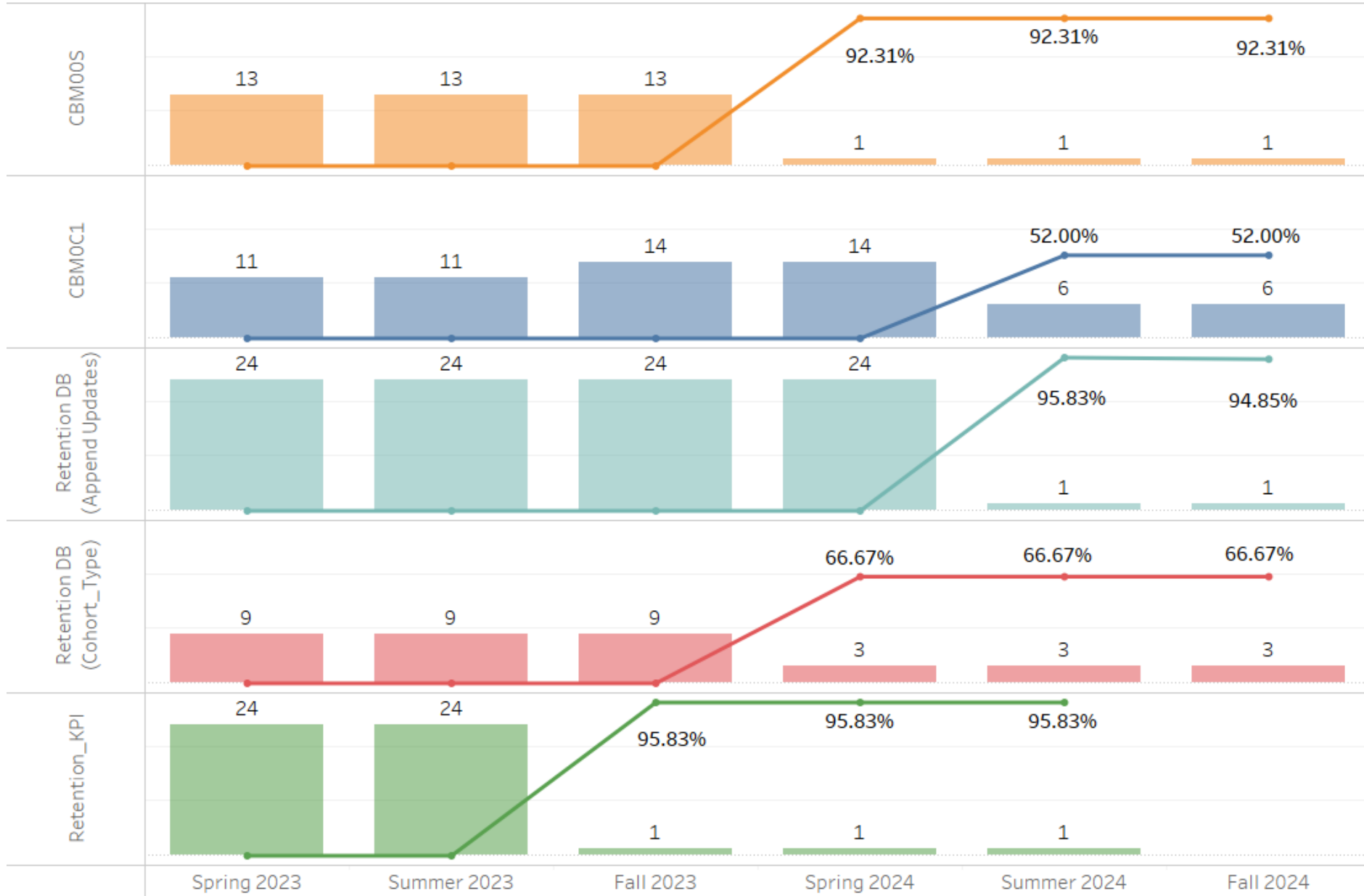
## Stored Procs & Dynamic SQL Impact on Code Base Size (Lines)





# IMPACT AND RESULTS

## Stored Procs & Dynamic SQL Impact on Data Quality Checks

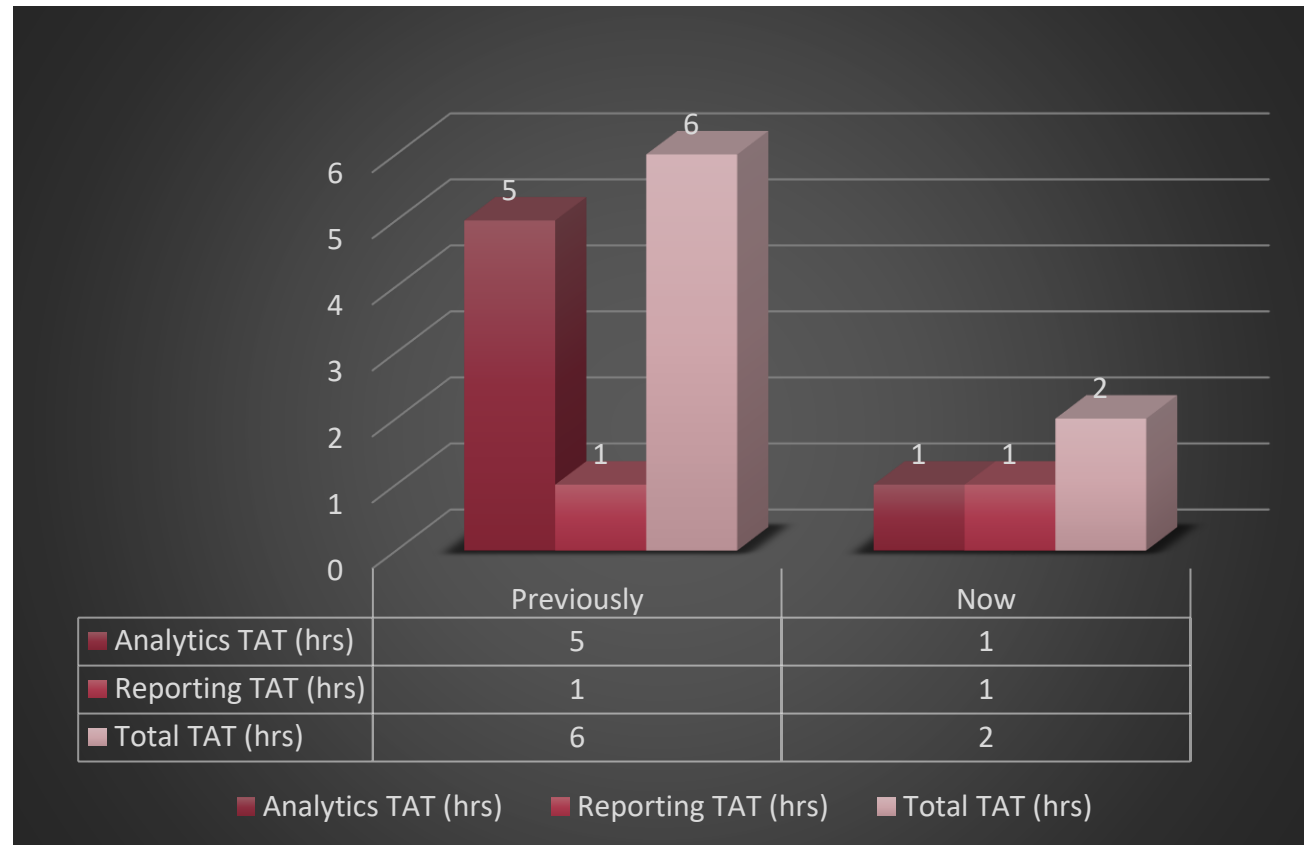


# IMPACT AND RESULTS

## HOW DOES IT HELP OTHER TEAMS @ PAIRS?

- **Faster Analytics = Higher Efficiency**
- Accelerated data processing improves overall team performance.
- Key Use Cases
  - Ad-hoc data requests
  - Reporting & data visualization
  - Tableau dashboards for leadership insights
  - Reduces delays in decision-making
  - Optimizes resource utilization & minimizes idle time
- Overall Benefit
  - Significant reduction in total TAT, leading to smoother & more agile operations

	Analytics TAT (hrs)	Reporting TAT (hrs)	Total TAT (hrs)	SpeedUp
Previously	5	1	6	
Now	1	1	2	300%





# Using GenAI Tools in SQL

# STATIC SQL TERM WISE UPDATES IN RETENTION DATABASE

```

UPDATE [dbo].[retention_final]
SET Term_2_Dept = cbm0c1_20239.dept,
    Term_2_Major = cbm0c1_20239.MAJR,
    Term_2_College = cbm0c1_20239.coll,
    Term_2_Program = cbm0c1_20239.PGM

FROM
[dbo].[cbm0c1_020239], [dbo].[retention_final]
WHERE [dbo].[retention_final].Term_1 = '20237'
AND cbm0c1_020239.PIDM =
[dbo].[retention_final].PIDM
AND cbm0c1_020239.Z_TERM = '20239'
AND cbm0c1_020239.TRF_FT = '';

UPDATE [dbo].[retention_final]
SET Term_2 = '20239'
WHERE Term_1 = '20237';
    
```

→ x 23 times (Term 2 to 24)

- We would almost have to manually change **Term number (Term\_2)**
- Manual update of **Term\_1** values
- Tedious manual work and **Time Taking for 24 different terms** and could also have human errors.

TERM_1	TERM_2	TERM_3	TERM_4	TERM_5	TERM_6	TERM_7	TERM_8
20239							
20237	20239						
20231	20237	20239					
20229	20231	20237	20239				
20227	20229	20231	20237	20239			
20221	20227	20229	20231	20237	20239		
20229	20221	20227	20229	20231	20237	20239	
20227	20229	20221	20227	20229	20231	20237	20239
20221	20227	20229	20221	20227	20229	20231	20237
						Z_TERM	Term_1

# PROMPTING AI

```
WHERE [dbo].[retention_final].Term_1 = @Term  
AND cbm0c1_020239.PIDM = [dbo].  
[retention_final].PIDM  
AND cbm0c1_020239.Z_TERM = '20239'  
AND cbm0c1_020239.TRF_FT = '';
```

```
UPDATE [dbo].[retention_final]  
SET Term_2 = '20239'  
WHERE Term_1 = '20237';
```

For the above SQL update statement for updates on term\_2, I want you to generate a SQL Update query for the table's columns for different terms (e.g., Term\_2, Term\_3, ..., Term\_24) and convert them into a dynamic SQL.

The SQL should loop from Term\_2 to Term\_24, dynamically computing the PrevTerm based on the following logic:

If the rightmost digit of @Term is 9 or 1, subtract 2 (e.g., 20249 becomes 20247, 20241 becomes 20239).  
If the rightmost digit is 7, subtract 6 (e.g., 20247 becomes 20241).

Note: Term\_1=@PrevTerm, don't change the 1 of term\_1

## Dynamic Term Update

```
1 DECLARE @i INT = 2;  
2 DECLARE @SQL NVARCHAR(MAX) = '';  
3 DECLARE @Term INT = 20249; -- Set this dynamically as needed  
4 DECLARE @PrevTerm INT;  
5  
6 WHILE @i <= 24  
7 BEGIN  
8     -- Compute PrevTerm based on the given logic  
9     IF RIGHT(@Term, 1) IN ('9', '1')  
10        SET @PrevTerm = @Term - 2;  
11     ELSE IF RIGHT(@Term, 1) = '7'  
12        SET @PrevTerm = @Term - 6;  
13     ELSE  
14        BREAK;  
15  
16     -- Construct the dynamic SQL for updating the columns  
17     SET @SQL = @SQL +  
18         'UPDATE [dbo].[retention_final] SET Term_' + CAST(@i AS NVARCHAR(2)) + '_Dept = cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '.dept, ' +  
19         'Term_' + CAST(@i AS NVARCHAR(2)) + '_Major = cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '.MAJR, ' +  
20         'Term_' + CAST(@i AS NVARCHAR(2)) + '_College = cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '.coll, ' +  
21         'Term_' + CAST(@i AS NVARCHAR(2)) + '_Program = cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '.PGM ' +  
22         'FROM [dbo].[cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '], [dbo].[retention_final] ' +  
23         'WHERE [dbo].[retention_final].Term_1 = '' ' + CAST(@PrevTerm AS NVARCHAR(6)) + '' ' +  
24         'AND cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '.PIDM = [dbo].[retention_final].PIDM ' +  
25         'AND cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '.Z_TERM = '' ' + CAST(@Term AS NVARCHAR(6)) + '' ' +  
26         'AND cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + '.TRF_FT = ''';  
27  
28     -- Update the term value for the next iteration  
29     SET @SQL = @SQL +  
30         'UPDATE [dbo].[retention_final] SET Term_' + CAST(@i AS NVARCHAR(2)) + ' = '' ' + CAST(@Term AS NVARCHAR(6)) + '' ' +  
31         'WHERE Term_1 = '' ' + CAST(@PrevTerm AS NVARCHAR(6)) + ''';  
32  
33     -- Prepare for the next iteration  
34     SET @Term = @PrevTerm;  
35     SET @i = @i + 1;  
36 END  
37  
38 -- Execute the dynamic SQL  
39 EXEC sp_executesql @SQL;  
40
```

Ask anything

+ Search Reason

# PROMPTING AI

ChatGPT Link: <https://chatgpt.com/share/67c00cb6-5a64-800d-97dd-be1e228aeaf2>

ChatGPT

Here is the dynamic SQL that will loop from `Term_2` to `Term_24` and update them based on the logic you provided.

Dynamic Term Update

This script dynamically generates and executes SQL updates for terms from `Term_2` to `Term_24`, calculating the previous term based on your logic. Let me know if you need any modifications!

I liked the SQL query you have generated, could you make it simpler and shorter by preserving the same logic

Edited Dynamic Term Update

I've simplified and shortened your SQL while maintaining the same logic. Let me know if you need any further refinements!



Dynamic Term Update

```
1 DECLARE @i INT = 2, @SQL NVARCHAR(MAX) = '', @Term INT = 20249, @PrevTerm INT;
2
3 WHILE @i <= 24
4 BEGIN
5     SET @PrevTerm = CASE WHEN RIGHT(@Term, 1) IN ('9', '1') THEN @Term - 2
6                       WHEN RIGHT(@Term, 1) = '7' THEN @Term - 6
7                       ELSE BREAK END;
8
9     SET @SQL += 'UPDATE [dbo].[retention_final] SET Term_' + CAST(@i AS NVARCHAR(2)) + '_Dept = c.dept, ' +
10              'Term_' + CAST(@i AS NVARCHAR(2)) + '_Major = c.MAJR, ' +
11              'Term_' + CAST(@i AS NVARCHAR(2)) + '_College = c.coll, ' +
12              'Term_' + CAST(@i AS NVARCHAR(2)) + '_Program = c.PGM ' +
13              'FROM [dbo].[cbm0c1_' + CAST(@Term AS NVARCHAR(6)) + ']' c, [dbo].[retention_final] r ' +
14              'WHERE r.Term_1 = ''' + CAST(@PrevTerm AS NVARCHAR(6)) + ''' ' +
15              'AND c.PIDM = r.PIDM AND c.Z_TERM = ''' + CAST(@Term AS NVARCHAR(6)) + ''' AND c.TRF_FT = '''';
16
17     SET @SQL += 'UPDATE [dbo].[retention_final] SET Term_' + CAST(@i AS NVARCHAR(2)) + ' = ''' + CAST(@Term AS NVARCHAR(6)) + ''' ' +
18              'WHERE Term_1 = ''' + CAST(@PrevTerm AS NVARCHAR(6)) + '''';
19
20     SET @Term = @PrevTerm;
21     SET @i += 1;
22 END
23
24 EXEC sp_executesql @SQL;
25
```

# PROMPTING AI

Before I execute the query would you explain me the query that you generated

Here's an explanation of the query:

## 1. Declaring Variables

```
sql Copy Edit
```

```
DECLARE @i INT = 2, @SQL NVARCHAR(MAX) = '', @Term INT = 20249, @PrevTerm INT;
```

- `@i` starts at `2` (since `Term_2` is the first one to update).
- `@SQL` is an empty string that will store the dynamically generated SQL statements.
- `@Term` is initialized to `20249` (the latest term).
- `@PrevTerm` will store the computed previous term.

## 2. Looping from Term\_2 to Term\_24

```
sql Copy Edit
```

Ask anything

+ Search Reason

## 7. Executing the Final Query

```
sql Copy Edit
```

```
EXEC sp_executesql @SQL;
```

- Executes the dynamically built SQL query.

## Summary

- Loops from `Term_2` to `Term_24`.
- Computes the previous term based on specific rules.
- Dynamically updates department, major, college, and program for each term.
- Updates `Term_X` in `retention_final`.
- Efficiently builds and executes the SQL statement dynamically.

Let me know if you need any refinements! 🚀

Copy Like Dislike Volume Share

# PROMPTING AI



I want to run this entire dynamic SQL in just one line of query, where I will call the UPDATE module and pass parameters like Term values, etc. Could you explain to me how I could achieve this?

To achieve this, you need to encapsulate the dynamic SQL logic into a **stored procedure**. This way, you can call it with parameters like `@StartTerm` and execute everything in a single query call.

## Steps to Achieve This

### 1. Create a Stored Procedure

Define a stored procedure that takes `@StartTerm` as input and performs the dynamic updates.

```
sql Copy Edit  
  
CREATE PROCEDURE UpdateRetentionTerms  
    @StartTerm INT  
AS  
BEGIN  
    DECLARE @i INT = 2, @SQL NVARCHAR(MAX) = '', @Term INT = @StartTerm, @PrevTerm INT;  
  
    WHILE @i <= 24  
    BEGIN  
        SET @PrevTerm = CASE  
            WHEN RIGHT(@Term, 1) IN ('9', '1') THEN @Term - 2  
            WHEN RIGHT(@Term, 1) = '7' THEN @Term - 6  
            ELSE BREAK  
        END
```

Ask anything



Search

Reason



### 2. Call the Procedure in One Line

Once the stored procedure is created, you can call it in just one line:

```
sql Copy Edit  
  
EXEC UpdateRetentionTerms @StartTerm = 20249;
```

## Explanation

#### 1. Stored Procedure ( `UpdateRetentionTerms` )

- Accepts `@StartTerm` as input.
- Loops through `Term_2` to `Term_24`.
- Calculates `@PrevTerm` dynamically.
- Builds an SQL string dynamically for `UPDATE` statements.
- Executes the SQL using `sp_executesql`.

#### 2. Calling the Procedure

- Just one SQL line: `EXEC UpdateRetentionTerms 20249;`
- No need to write dynamic SQL every time.
- Easy to reuse for different terms.

## Key Benefits

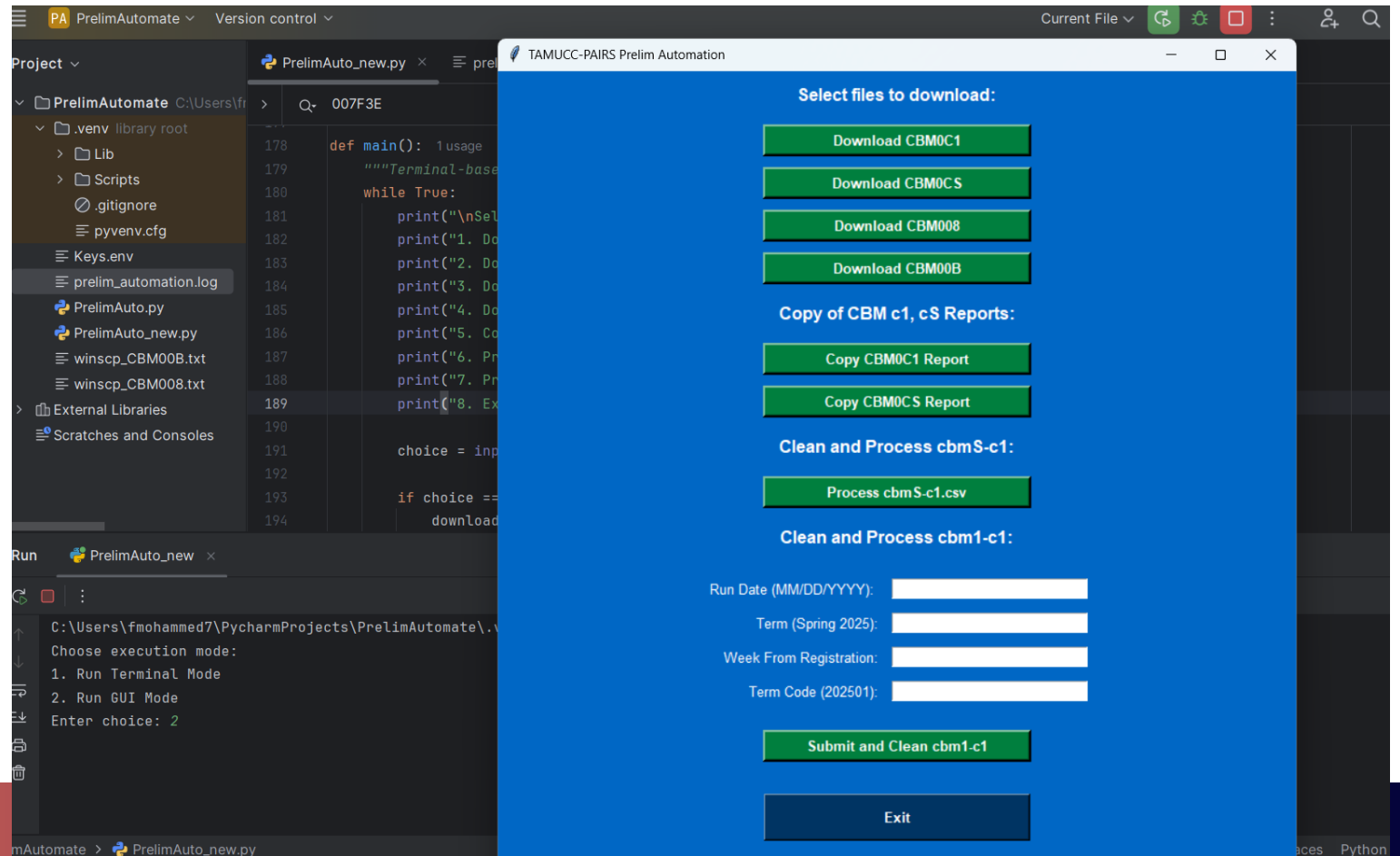
- ✓ **Simplifies Execution** – One-line execution instead of writing a long SQL query.
- ✓ **Encapsulates Logic** – The procedure handles all updates internally.
- ✓ **Reusable** – Call it anytime with a different term.
- ✓ **Scalable** – If logic changes, update the procedure instead of modifying queries everywhere.

Would you like me to refine this further or add any additional functionality? 🚀



# AUTOMATING CBM FILE TRANSFERS & PROCESSING

- **Challenge:** Manually downloading and cleaning the CBM files from Banner (or any server) and organizing them into network folders based on report types and dates was time-consuming and error-prone.
- **Solution:** We automated the process using a GenAI tool like ChatGPT to develop a Python script, that provides a custom GUI that solves our challenge with just one-click.



# PROMPTING THE RIGHT THING!

## Prompt 1 (User's Unclear Input in Simple English)

*"I have two tables with student data, and I want to join them together to see all student info. Also, I need to update some records in both tables. How can I do this?"*

## Prompt 2 (Refined Version)

*"I have two SQL tables containing student data: one stores general student details (e.g., `students` with `student_id`, `name`, `email`), and the other contains academic records (e.g., `grades` with `student_id`, `course`, `grade`). I want to:*

- 1. Perform an SQL join to merge relevant student details with their grades.*
- 2. Update specific records in both tables, such as modifying a student's email in `students` and updating a grade in `grades`.*

*Can you provide SQL queries to achieve this?"*

# LEARNING RESOURCES



[Explore Courses](#)

[AI Newsletter](#) ▾

[🌟 AI Dev 25](#)

[Community](#) ▾

[Company](#) ▾

[Log In](#)

[All Courses](#) > [Short Courses](#) > [ChatGPT Prompt Engineering for Developers](#)

Short Course

Beginner

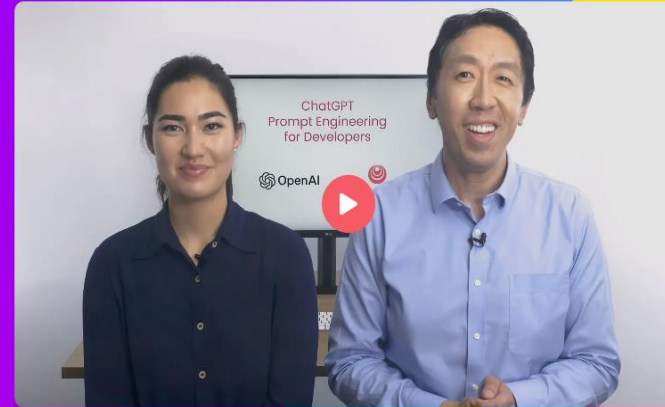
1 Hour 30 Minutes

## ChatGPT Prompt Engineering for Developers

Instructors: Isa Fulford, Andrew Ng



Learn for Free



← ↻ 🔒 <https://www.promptingguide.ai>

**Prompt Engineering Guide**

[Prompt Engineering Course](#) [Services](#) [About](#)

**Prompt Engineering**

Introduction ▾

LLM Settings

Basics of Prompting

Prompt Elements

General Tips for Designing

Prompt Engineering

## Prompt Engineering Guide

Prompt engineering is a relatively new discipline for developing and optimizing prompts to efficiently use language models (LLMs) for a wide variety of applications and research topics. Prompt engineering skills help to better understand the capabilities and limitations of large language models (LLMs).

## LEVERAGING GENAI TOOLS (CHATGPT)

- **Used ChatGPT for:**
  - **Upskilling team in advanced SQL techniques.**
  - **Crafting complex SQL queries via Text-to-SQL prompts.**
- **Benefits of Generative AI:**
  - **Time-saving SQL query generation.**
  - **Improved problem-solving and SQL brainstorming.**

## BEST PRACTICES FOR USING AI TOOLS

- **Carefully manage sensitive data when using AI tools.**
- **Verify AI-generated queries before executing.**
- **Use AI for learning and enhancing SQL skills, not as a sole solution**

## KEY TAKEAWAYS

- **Stored Procedures and Dynamic SQL are essential for efficient database management**
- **GenAI tools like ChatGPT are valuable for SQL learning and automation support**
- **Automation boosts productivity and minimizes human errors**

# CONCLUSION

- **Conclusion:**

- Stored Procedures and Dynamic SQL improved speed, accuracy, and automated repetitive tasks.
- Dynamic SQL ensures flexibility for complex validations and multi-term data management.
- GenAI tools like ChatGPT helped us upskill and develop advanced SQL queries with Text-to-SQL prompting.

- **Why Not Use Stored Procedures for All Updates:**

- Granular Reviews: Some updates need detailed checks by DB admins due to dynamic data.
- Maintenance: Stored Procedures can be harder to maintain with changing data models.
- Not portable. If you change your database engine you have to scrap and redo them.
- Testing? Nightmare.

## FUTURE PLANS

- Continue integrating Stored Procedures into additional workflows to enhance automation.
- Explore more uses of GenAI tools for advanced automation and query generation.
- Focus on continuous improvement and knowledge sharing to maintain efficiency and enhance team expertise



Q & A





# THANK YOU!

Farazuddin Mohammed  
Graduate Assistant - Research  
[fmohammed7@islander.tamucc.edu](mailto:fmohammed7@islander.tamucc.edu)

Jiashi Zhao, EDD  
Lead Data Analyst-Analytics  
[Jiashi.zhao@tamucc.edu](mailto:Jiashi.zhao@tamucc.edu)

Scan the QR code to  
complete the session  
survey.



Texas Association for Institutional Research

Annual Conference: February 25-28, 2025

Omni Hotel in Corpus Christi, TX

