Systems Design & Databases

Transact - SQL

SQL Server - TSQL Queries to support :

Safety At Work

Name : Hina Malik

Course : Systems Design and databases

Date : 30th May 2025

Tutor Name : Miss Bianca Ogbo

SQL Server - TSQL Queries to support :

Safety At Work

Table of Appendices

SQL Server Practitioner Details:

SQL Server - TSQL Practitioner Details:		
	**Name:**	**Hina Malik**
	Email:	**Hinamalik03045@gmail.com**
	Course:	**Systems Design and Databases**
	Date:	**30/04/2025**
	Tutor:	**Miss Bianca Ogbo**

a. Introduction to the SQL Practitioner:

Why you decided upon studying and pursuing your interests in becoming a graduate developer.

As a junior developer with a strong interest in system design and database management, I am passionate about using technology to create meaningful and impactful solutions. My journey into development began with my fascination for structured workflows and problem-solving, which naturally led me to focus on tools like SQL for efficient data handling and insights.

Pursuing a career as a graduate developer aligns with my strengths in logical thinking, adaptability, and attention to detail. I decided to study this field because of its dynamic nature and the potential it offers to transform industries, particularly through the power of data and software development. My goal is to contribute to innovative projects that enhance user experiences and streamline operations, combining my technical skills with creativity and a user-centered approach.

b. **Why you should learn SQL:**

Learning SQL (Structured Query Language) is crucial for anyone pursuing a career in software development, data science, or machine learning. SQL Server is widely used across industries for its scalability and security, making it a valuable skill that enhances your employability.

SQL is essential for managing and analyzing data, which is vital for informed decision-making in business. By mastering SQL, you can efficiently access and manipulate data, allowing you to unlock insights that drive innovation.

I recommend learning SQL because it has universal applications in fields like healthcare, finance, and AI. For my career, I aim to gain confidence in designing data systems and contribute to meaningful projects that bridge the gap between data and applications.

Graduate Developer vacancy SQL server

https://www.linkedin.com/jobs/view/4183961618

https://www.totaljobs.com/job/data-analyst-graduate-peterborough/circle-group-job104721155

https://www.milkround.com/job/entry-level-data-engineer/outsource-uk-ltd-job104796442

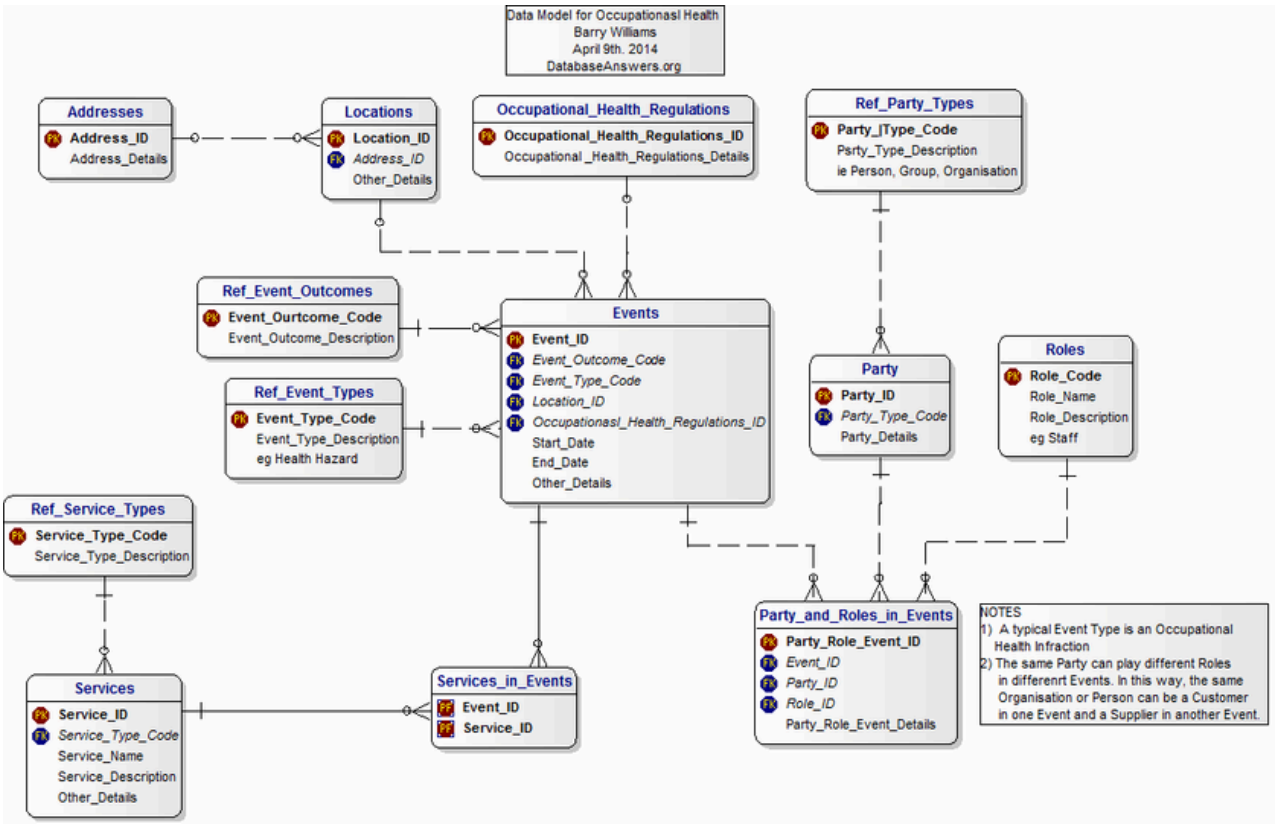
SQL Server Database Overview:

a. **SQL Server Database for Demos:** Safety At Work

I am using the Safety At Work SQL Server database, which stores data on workplace accidents, employee safety training, inspections, and incident reports. My goal is to create useful TSQL queries and scripts that support business needs and user requirements.

These queries provide valuable insights for front-end applications (web or mobile), such as tracking accidents, monitoring training, and generating safety reports. Below are examples of my best TSQL demos designed to help improve workplace safety and decision-making.

b. SQL Server Database Diagrams:



SQL Practitioners TSQL Demos:

Introduction:

Below is an audit trail of my best TSQL query examples, designed to meet both business and user requirements within the Safety At Work database. These demos illustrate how my SQL skills support key business functions—such as compliance reporting and safety analysis—while also delivering meaningful, user-focused insights like tracking training progress and monitoring incidents. Each script demonstrates my ability to create practical solutions that benefit both the organization and its end users.

TSQL Part 1: SQL Server Coding Basics

This covers ALL the basic TSQL skills from Modules 01 to 09. This will help new SQL Practitioner to become familiar with the basic SQL coding skills.

1. TSQL03 to TSQL08: SQL Server Basics

This section covers the basics skills in using SSMS and scoping TSQL Queries either by code or by using the Design Query in Editor, on how to use Select statements to query data from table(s), Join across related tables, sort and filtering with Where, modifying data and using built in functions for Safety At Work:

.sql File for

TSQL01-09 Demos:	https://drive.google.com/file/d/1lnns54A62M5rzA7-C3ia6wI4VI8kXVGj/view

a. Module 3: Writing SELECT Queries with Single Table

Why write Select queries?

The SELECT statement serves to query database tables and apply logical operations to the data, producing a set of results.

Demo A1: Writing a Simple SELECT Statement:

1.

Retrieve Events and Their Outcomes

"As a Health and Safety Officer, I want to see all events along with their outcomes so that I can track compliance with workplace regulations. "

Select Statement Query :

```
SELECT * FROM Events;
```

Results :

Results Messages

	Event_ID	Event_Outcome_Code	Event_Type_Code	Location_ID	Occupational_Health_Regulations_ID	Start_Date	End_Date	Other_Details
1	1	3	11	15	1	2024-07-26	2025-01-01	Includes Hands-On training Session
2	2	3	7	4	7	2024-04-18	2025-01-01	On-Session at Main Office
3	3	6	6	10	5	2024-08-22	2025-01-01	Mandatory for all Employees
4	4	16	8	1	3	2025-01-11	2025-01-01	Includes Hands-On training Session
5	5	13	6	6	14	2024-12-23	2025-01-01	Includes Hands-On training Session
6	6	3	2	11	13	2025-03-14	2025-01-01	Follow-Up Required after Completion
7	7	9	14	5	5	2024-05-11	2025-01-01	Mandatory for all Employees
8	8	1	13	12	16	2024-07-11	2025-01-01	Held Virtually Via Teams
9	9	11	8	3	4	2025-01-20	2025-01-01	Mandatory for all Employees
10	10	14	15	15	5	2024-05-03	2025-01-01	Includes Hands-On training Session
11	11	13	13	6	8	2024-08-31	2025-01-01	Follow-Up Required after Completion
12	12	14	5	6	13	2024-12-25	2025-01-01	Follow-Up Required after Completion
13	13	6	14	11	1	2024-08-01	2025-01-01	Includes Hands-On training Session
14	14	2	6	14	11	2024-06-02	2025-01-01	Mandatory for all Employees
15	15	14	13	5	10	2024-10-17	2025-01-01	Held Virtually Via Teams
16	16	9	10	14	7	2024-11-12	2025-01-01	On-Session at Main Office

Explanation :

- **SELECT** : Specifies which columns of data to retrieve from the database.
- **" * "** : Means "all columns"—it tells SQL to return every column in the table.
- **FROM** : Indicates which table to retrieve the data from.
- **Events** : The name of the table containing the event data.

Insights :

- **Process Improvement** : The query supports detailed analysis of event characteristics and outcomes, facilitating data-driven decisions.
- **Increased Efficiency** : It allows access to comprehensive event data, which aids in identifying trends and areas for optimization, leading to increased efficiency.

1a.

Retrieve Start Date of All Events for Health and Safety Scheduling

" As a Health and Safety Officer, I want to see the Start Date of all events so that I can ensure the proper scheduling of safety-related activities."

Select Statement Query :

```
SELECT Start_Date FROM Events;
```

Results:

Results Messages

	Start_Date
1	2024-07-26
2	2024-04-18
3	2024-08-22
4	2025-01-11
5	2024-12-23
6	2025-03-14
7	2024-05-11
8	2024-07-11
9	2025-01-20

Explanation :

- SELECT: Specifies which column(s) to retrieve from the database.
- Start_Date: The specific column you want to retrieve, which contains the start dates of events.
- FROM: Indicates the table to retrieve the data from.
- Events: The name of the table where the event data is stored.

Insights :

- **Minimizer Risks** : This query enables efficient planning and prioritization of safety-related events, ensuring compliance with workplace regulations and minimizing risks.
- **Enhances Scheduling Accuracy** : Improves event tracking by providing clear start dates for better planning and resource allocation.

Demo A2: Eliminating Duplicates with DISTINCT

1.

Retrieve Unique Service Names

"As a Training Coordinator, I want to see all unique services provided so that I can plan future training programs."

Distinct Clause Query :

```
SELECT DISTINCT Service_Name FROM Services;
```

Results :

Before `DISTINCT` command :

Results Messages

	Services_ID	Service_Type_Code	Service_name	service_Description	Other_Details
1	1	16	Personal Training	Virtual assistant services for small businesses	Availability Online and in-person
2	2	1	Event Planning	Fitness coaching and personalized workout plans	Offered Quarterly
3	3	10	Event Planning	Mobile car detailing and wash services	Offered Quarterly
4	4	12	Event Planning	Tech support and IT consulting for individuals a...	Auto Follow-Up after completion.
5	5	14	Landscaping	Professional cleaning services for residential pro...	Optional for all Employees
6	6	4	Pet Grooming	Professional cleaning services for residential pro...	Conducted Yearly
7	7	12	Home Renovation	Tech support and IT consulting for individuals a...	Optional for all Employees
8	8	11	Home Renovation	Mobile car detailing and wash services	Requires Early Registration
9	9	13	Pet Grooming	Graphic design and branding services for startups	Optional for all Employees
10	10	6	Massage Therapy	Landscaping and lawn care maintenance	Optional for all Employees
11	11	14	Cleaning Services	Professional cleaning services for residential pro...	Conducted Yearly
12	12	14	Event Planning	Graphic design and branding services for startups	Requires Early Registration
13	13	6	Massage Therapy	Landscaping and lawn care maintenance	Requires Early Registration
14	14	1	Graphic Design	Interior design consultation and home staging s...	Availability Online and in-person
15	15	10	Home Renovation	Fitness coaching and personalized workout plans	Optional for all Employees
16	16	6	Event Planning	Virtual assistant services for small businesses	Conducted Yearly

After `DISTINCT` command :

Results Messages

	Service_Name
1	Cleaning Services
2	Event Planning
3	Graphic Design
4	Home Renovation
5	Landscaping
6	Massage Therapy
7	Personal Training
8	Pet Grooming

Explanation :

- **SELECT**: Specifies which column(s) to retrieve from the database.
- **DISTINCT**: Ensures that only unique (non-duplicate) values are returned.
- **Service_Name**: The specific column you want to retrieve, which contains the names of services.
- **FROM**: Indicates the table to retrieve the data from.
- **Services**: The name of the table where the service data is stored.

Insights :

- **Efficient Planning**: The query offers a clear overview of available services, supporting comprehensive coverage and effective resource allocation for training programs.
- **Targeted Training**: By revealing the range of services, the query enables training initiatives to be aligned with business needs, optimizing employee skill development.

1a.

Retrieve Unique Location Details

" As a facility manager, I want to retrieve a distinct list of location details from the Locations table, so that I can efficiently understand the variety of location types available for events or activities and optimize space management."

Distinct Clause Query :

```
SELECT DISTINCT Other_Details FROM Locations;
```

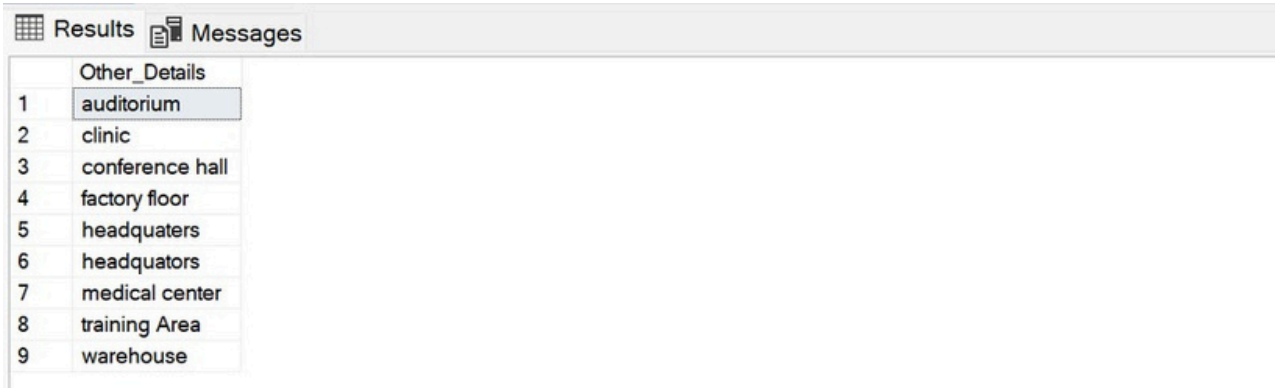
Results:

Before `DISTINCT` command

Results Messages

	Location_ID	Address_ID	Other_Details
1	1	13	headquators
2	2	7	factory floor
3	3	7	training Area
4	4	9	medical center
5	5	5	conference hall
6	6	6	warehouse
7	7	8	conference hall
8	8	3	factory floor
9	9	10	training Area
10	10	2	clinic
11	11	16	headquaters
12	12	8	training Area
13	13	3	training Area
14	14	15	conference hall
15	15	10	auditorium
16	16	5	factory floor

After `DISTINCT` command :



Results | Messages

	Other_Details
1	auditorium
2	clinic
3	conference hall
4	factory floor
5	headquaters
6	headquators
7	medical center
8	training Area
9	warehouse

Explanation :

- **SELECT**: Specifies which column(s) to retrieve from the database.
- **DISTINCT**: A keyword that ensures only unique (non-duplicate) values are returned.
- **Other_Details:** The specific column from which you want to retrieve unique values, such as "conference hall", "training area", "auditorium", and "factory floor."
- **FROM**: Indicates the table to retrieve the data from.
- **Locations:** The name of the table which includes location data.

Insights :

- **Clear Overview**: This query helps optimize space management by providing a clear list of available location types for events or activities.
- **Supports Efficient Space Utilization:** Understanding the variety of location options supports efficient distribution of resources and planning.

Demo A3: Using Column and Table Aliases Lesson:

1.

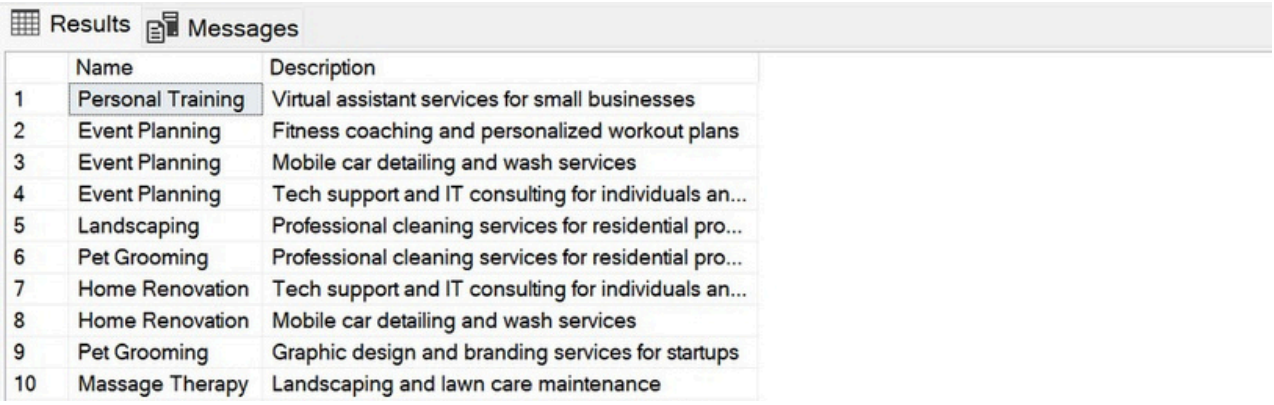
Retrieve Service Names and Descriptions from the Database

“ As a Database Analyst, I want to retrieve service names and their descriptions so that I can present a clear list of available services for reporting, easily.”

As keyword Query :

```
SELECT
       S.Service_Name AS Name,
       S.Service_Description AS Description

FROM   Services AS S;
```

Results :



	Name	Description
1	Personal Training	Virtual assistant services for small businesses
2	Event Planning	Fitness coaching and personalized workout plans
3	Event Planning	Mobile car detailing and wash services
4	Event Planning	Tech support and IT consulting for individuals an...
5	Landscaping	Professional cleaning services for residential pro...
6	Pet Grooming	Professional cleaning services for residential pro...
7	Home Renovation	Tech support and IT consulting for individuals an...
8	Home Renovation	Mobile car detailing and wash services
9	Pet Grooming	Graphic design and branding services for startups
10	Massage Therapy	Landscaping and lawn care maintenance

Explanation :

- **SELECT S.Service_Name AS Name, S.Service_Description AS Description :** Retrieves the Service_Name and Service_Description columns from the table.
- **S.Service_Name AS Name :** Selects the Service_Name column and renames it to Name in the result set using a column alias (AS Name)
- **S.Service_Description AS Description :** Selects the Service_Description column and renames it to Description in the result set using a column alias (AS Description)
- **FROM Services AS S :** Specifies the Services table as the data source and assigns it a table alias S, which is used to qualify column names in the SELECT clause

Insights:

- **Improves Service Reporting** : This query is useful for generating service catalogs or summaries for users or stakeholders and provides a clear list of services for easy reference by users and stakeholders.
- **Supports Better Decision-Making:** Structured service details help align planning with business goals.

1a.

Ensuring Safety Standards

“ As a compliance officer, I need to retrieve all occupational health regulations details from the Occupational_Health_Regulations table, so that I can review and ensure the company's adherence to the health and safety standards, thereby reducing legal and operational risks. ”

As keyword Query :

```
SELECT O.Occupational_Health_Regulations_Details AS Details,

FROM Occupational_Health_Regulations AS O;
```

Results :

Results | Messages

	Details
1	Regular breaks must be taken every 2 hours
2	Training on hazardous materials handling must be pr...
3	First aid kits must be fully stocked and easily accessible
4	Fire drills must be conducted every quarter
5	Regular ergonomic assessments must be conducted ...
6	Proper ventilation must be maintained in work areas
7	Employees must undergo annual health screenings
8	Regular ergonomic assessments must be conducted ...
9	First aid kits must be fully stocked and easily accessible
10	Proper lifting techniques must be followed to prevent i...
11	Employees must wear safety goggles at all times in de...
12	Emergency exits must be clearly marked and accessi...
13	Employees must wear safety goggles at all times in de...
14	Training on hazardous materials handling must be pr...
15	Proper lifting techniques must be followed to prevent i...
16	Employees must wear safety goggles at all times in de...

Explanation :

- **SELECT DISTINCT O.Occupational_Health_Regulations_Details AS Details :** Retrieves only unique entries from the Occupational_Health_Regulations_Details column and renames the output column as Details using a column alias.
- **FROM Occupational_Health_Regulations AS O** : Specifies the table to query and assigns it a table alias O for easier reference.

Insights :

- **Supports Compliance Review:** Provides a clear, non-duplicated list of occupational health regulations, making it easier for compliance officers to review all relevant standards.
- **Reduces Risk:** Access to unique regulation details helps ensure adherence to health and safety requirements, reducing legal and operational risks for the company.

Demo A4: Writing Simple CASE Expressions

1.

Categorize Events Based on Duration

“ As a Health and Safety Officer, I want to categorize events based on their duration so that I can identify long-running events.”

Case Expression Query :

```
SELECT Event_ID,
    CASE
        WHEN DATEDIFF(DAY, Start_Date, End_Date) <= 1 THEN 'Short'
        WHEN DATEDIFF(DAY, Start_Date, End_Date) BETWEEN 2 AND 5
        THEN 'Medium'
        ELSE 'Long'
    END AS Event_Duration_Category
FROM Events;
```

Results :

Before `CASE` command,

Results Messages

	Event_ID	Event_Outcome_Code	Event_Type_Code	Location_ID	Occupational_Health_Regulations_ID	Start_Date	End_Date	Other_Details
1	1	3	11	15	1	2024-07-26	2025-01-01	Includes Hands-On training Session
2	2	3	7	4	7	2024-04-18	2025-01-01	On-Session at Main Office
3	3	6	6	10	5	2024-08-22	2025-01-01	Mandatory for all Employees
4	4	16	8	1	3	2025-01-11	2025-01-01	Includes Hands-On training Session
5	5	13	6	6	14	2024-12-23	2025-01-01	Includes Hands-On training Session
6	6	3	2	11	13	2025-03-14	2025-01-01	Follow-Up Required after Completion
7	7	9	14	5	5	2024-05-11	2025-01-01	Mandatory for all Employees
8	8	1	13	12	16	2024-07-11	2025-01-01	Held Virtually Via Teams
9	9	11	8	3	4	2025-01-20	2025-01-01	Mandatory for all Employees
10	10	14	15	15	5	2024-05-03	2025-01-01	Includes Hands-On training Session

After `CASE` command,

Results | Messages

	Event_ID	Event_Duration_Category
1	1	Long
2	2	Long
3	3	Long
4	4	Short
5	5	Long
6	6	Short
7	7	Long
8	8	Long
9	9	Short
10	10	Long

Explanation :

- **SELECT Event_ID, ... :** Retrieves the Event_ID and the computed event duration category.
- **CASE ... END AS Event_Duration_Category :** Uses conditional logic to label each event as 'Short', 'Medium', or 'Long' based on the number of days between Start_Date and End_Date.
- **FROM Events; :** Specifies the source table.

Insights :

- **Targeted Attention :** Helps identify long events needing extra focus and resources.
- **Prioritized Safety :** Supports prioritizing safety measures based on event duration.

1a.

Manage Services with Clarity

“ As a service manager, I want to view specific details related to event planning and home renovation services alongside general service descriptions, so that I can quickly access relevant information for different service types.”

Case Expression Query:

```
SELECT Services_ID,
      CASE
         WHEN Service_name = 'event planning' THEN other_Details
         WHEN Service_name = 'home renovation' THEN other_Details
         ELSE service_Description
      END AS events_in_consideration
FROM services;
```

Results :

Before `CASE` command,

Results Messages

	Services_ID	Service_Type_Code	Service_name	service_Description	Other_Details
1	1	16	Personal Training	Virtual assistant services for small businesses	Availability Online and in-person
2	2	1	Event Planning	Fitness coaching and personalized workout plans	Offered Quarterly
3	3	10	Event Planning	Mobile car detailing and wash services	Offered Quarterly
4	4	12	Event Planning	Tech support and IT consulting for individuals an...	Auto Follow-Up after completion.
5	5	14	Landscaping	Professional cleaning services for residential pro...	Optional for all Employees
6	6	4	Pet Grooming	Professional cleaning services for residential pro...	Conducted Yearly
7	7	12	Home Renovation	Tech support and IT consulting for individuals an...	Optional for all Employees
8	8	11	Home Renovation	Mobile car detailing and wash services	Requires Early Registration
9	9	13	Pet Grooming	Graphic design and branding services for startups	Optional for all Employees
10	10	6	Massage Therapy	Landscaping and lawn care maintenance	Optional for all Employees
11	11	14	Cleaning Services	Professional cleaning services for residential pro...	Conducted Yearly
12	12	14	Event Planning	Graphic design and branding services for startups	Requires Early Registration
13	13	6	Massage Therapy	Landscaping and lawn care maintenance	Requires Early Registration
14	14	1	Graphic Design	Interior design consultation and home staging se...	Availability Online and in-person
15	15	10	Home Renovation	Fitness coaching and personalized workout plans	Optional for all Employees
16	16	6	Event Planning	Virtual assistant services for small businesses	Conducted Yearly

After `CASE` command,

Results Messages

	services_iD	events_in_consideration
1	1	Virtual assistant services for small businesses
2	2	Offered Quarterly
3	3	Offered Quarterly
4	4	Auto Follow-Up after completion.
5	5	Professional cleaning services for residential pro...
6	6	Professional cleaning services for residential pro...
7	7	Optional for all Employees
8	8	Requires Early Registration
9	9	Graphic design and branding services for startups
10	10	Landscaping and lawn care maintenance
11	11	Professional cleaning services for residential pro...
12	12	Requires Early Registration
13	13	Landscaping and lawn care maintenance
14	14	Interior design consultation and home staging se...
15	15	Optional for all Employees
16	16	Conducted Yearly

Explanation :

- **SELECT Services_ID, ...** : Retrieves the unique identifier for each service.
- **CASE ... END AS events_in_consideration** : Uses a CASE expression to determine what information to display in the events_in_consideration column:
  If Service_name is 'event planning' or 'home renovation', it displays the value from other_Details. For all other service names, it displays the value from service_Description. The result is aliased as events_in_consideration for clarity.
- **FROM services;** : Specifies the source table containing service data.

Insights :

- **Focused Information** : Displays tailored details for important services while providing general descriptions for others, improving clarity.
- **Easier Evaluation** : Helps stakeholders quickly review and compare services in a clear, organized format.

Demo A5: Writing Aggregate Functions:

1.

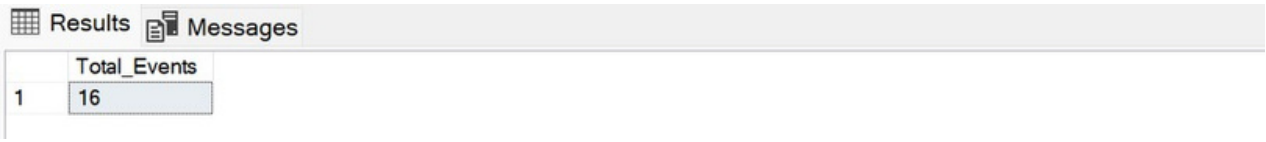
Total Event Count

" As a Health and Safety Officer, I want to analyze event data using aggregate functions so that I can gain insights into the total number of events, their durations, and other key metrics."

1. COUNT Function Query:

```
SELECT COUNT (Event_ID) AS Total_Events FROM Events;
```

Results :



Explanation :

- SELECT COUNT(Event_ID) AS Total_Events
  a. COUNT(Event_ID): Counts the number of non-null Event_ID entries in the Events table,
    - representing the total number of events.

  b. AS Total_Events: Assigns a clear, descriptive alias to the result column for easy interpretation.
- FROM Events;
  a. Specifies the Events table as the source of the data being counted.

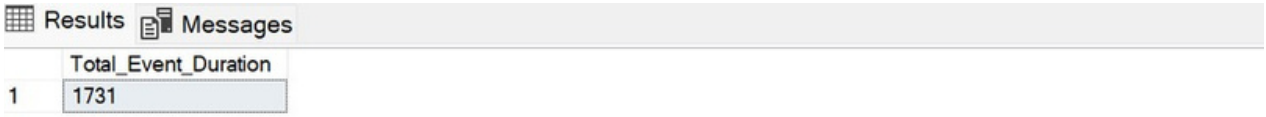
Insights :

- **Workload Assessment** : Knowing there are 16 events helps the Health and Safety Officer gauge the current workload and allocate resources effectively.
- **Trend Monitoring** : Tracking the total number of events over time reveals patterns that can inform adjustments to safety protocols and staffing needs.

2. SUM Function Query :

```
SELECT SUM (DATEDIFF (DAY, Start_Date, End_Date))
AS Total_Event_Duration FROM Events;
```

Results :



Explanation :

- **SELECT SUM(DATEDIFF(DAY, Start_Date, End_Date)) AS Total_Event_Duration**

a. DATEDIFF(DAY, Start_Date, End_Date): Calculates the duration in days for each event.
b. SUM(...): Adds up all individual event durations to get the total number of days.
c. AS Total_Event_Duration: Labels the result column for clarity.

- **FROM Events;**

a. Specifies the Events table as the source of the data.

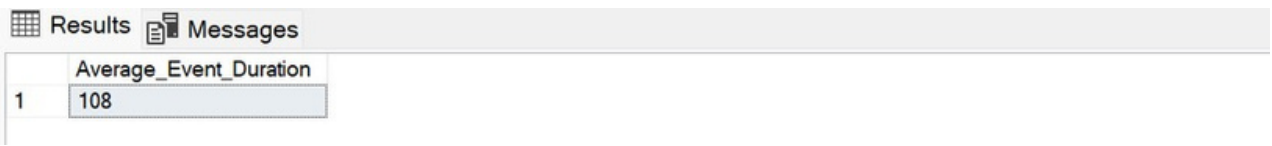
Insights :

- **Operational Insight**
  Summing total event durations helps organizations optimize scheduling and gain a clearer understanding of their operational timelines.
- **Resource Planning**
  Knowing that all events together lasted 1,731 days enables businesses to assess overall time investment and allocate resources more efficiently.

3. AVG Function Query :

```
SELECT AVG (DATEDIFF (DAY, Start_Date, End_Date))
AS Average_Event_Duration FROM Events;
```

Results :



Explanation :

- **SELECT AVG(DATEDIFF(DAY, Start_Date, End_Date)) AS Average_Duration**
  a. DATEDIFF(DAY, Start_Date, ...): Calculates the duration in days for each event.
  b. AVG(...): Computes the average of all event durations.
  c. AS Average_Duration: Names the result column for easy understanding.
- **FROM Events;**
  a. Specifies the Events table as the source of the data.

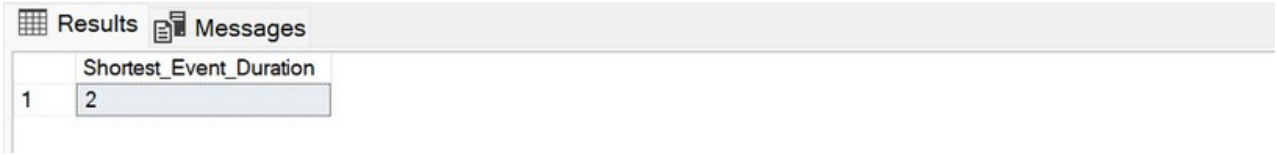
Insights :

- **Scheduling Optimization**
  An average event duration of 108 days helps businesses set realistic timelines, improving scheduling and resource planning.
- **Strategic Planning**
  Understanding typical event length enables better marketing strategies and enhances overall efficiency in event management.

4. MIN Function Query :

```
SELECT MIN
      (CASE WHEN Start_Date <= End_Date THEN DATEDIFF
      (DAY, Start_Date, End_Date) ELSE NULL END)
AS Shortest_Event_Duration FROM Events;
```

Results :



Explanation :

- **SELECT MIN(CASE WHEN Start_Date <= End_Date THEN DATEDIFF(DAY , Start_Date, End_Date) ELSE NULL END) AS Shortest_Event_Duration**
  a. **CASE WHEN ... ELSE NULL END**: Filters out invalid records where Start_Date is after End_Date (ensuring logical date ranges).
  b. **DATEDIFF(DAY, ...)**: Calculates event duration in days for valid entries.
  c. **MIN(...)**: Identifies the shortest duration from the valid results.
- **FROM Events**: Pulls data from the Events table.

Insights :

- **Efficiency Opportunities**
  Identifies the shortest event duration, highlighting opportunities for quick-turnaround activities that reduce resource strain and operational costs.
- **Data Integrity Assurance**
  Filters out illogical date entries (e.g., start dates after end dates), ensuring accurate analysis and reliable planning for future events.

5. MAX Function Query :

```
SELECT MAX (DATEDIFF (DAY, Start_Date, End_Date))

AS Longest_Event_Duration FROM Events;
```

Results :

Explanation :

- **SELECT MAX(DATEDIFF(DAY, Start_Date, End_Date)) AS Longest_Duration**
  a. DATEDIFF(DAY, Start_Date, End_Date): Calculates the number of days between the start and end dates for each event.
  b. MAX(...): Finds the maximum value among all calculated durations, representing the longest event.
  c. AS Longest_Duration: Assigns a clear, descriptive name to the result column.
- **FROM Events;**
  a. Specifies the Events table as the source of the data.

Insights :

- **Commitment Estimation**
  Knowing that an event lasted 258 days helps teams anticipate the level of dedication and planning required for similar long-term projects.
- **Resource Management**
  Highlighting such lengthy events allows for better allocation and monitoring of resources to ensure successful project completion.

b. Module 4: Joining and Querying Multiple Tables

Why use Joining and Querying Multiple Tables?

Joining multiple tables lets you bring together related information stored in different places, so you can see everything you need in one result. This makes your data easier to understand and helps you find useful connections between different types of information.

Demo B1: How to provide data from 2 or more related tables with multiple or singular Join statements

1.

Querying with 2 or Multiple Tables

" As a Health and Safety Officer, I want to retrieve event details along with their associated services and locations so that I can track compliance and logistics. "

1. INNER JOIN Statement Query :

```
SELECT E.Event_ID AS "Event ID",
       E.Start_Date AS "Start Date",
       E.End_Date AS "End Date",
       S.Service_Name AS "Service Name",
       L.Other_Details AS "Location Name"
FROM Events AS E
 INNER JOIN Services_In_Events AS SE ON E.Event_ID = SE.Event_ID
 INNER JOIN Services AS S ON SE.Service_ID = S.Services_ID
 INNER JOIN Locations AS L ON E.Location_ID = L.Location_ID;
```

Results :

Results Messages

	Event ID	Start Date	End Date	Service Name	Location Name
1	1	2024-07-26	2025-01-01	Personal Training	auditorium
2	2	2024-04-18	2025-01-01	Event Planning	medical center
3	3	2024-08-22	2025-01-01	Event Planning	clinic
4	4	2025-01-11	2025-01-01	Event Planning	headquators
5	5	2024-12-23	2025-01-01	Landscaping	warehouse
6	6	2025-03-14	2025-01-01	Pet Grooming	headquaters
7	7	2024-05-11	2025-01-01	Home Renovation	conference hall
8	8	2024-07-11	2025-01-01	Home Renovation	training Area
9	9	2025-01-20	2025-01-01	Pet Grooming	training Area
10	10	2024-05-03	2025-01-01	Massage Therapy	auditorium

Explanation :

- **SELECT ...**
  Retrieves event ID, start and end dates, service name, and location name, each with a user-friendly alias.
- **FROM Events AS E**
  Starts with the Events table, given the alias E.
- **INNER JOIN Services_In_Events AS SE ON E.Event_ID = SE.Event_ID**
  Joins with the Services_In_Events table to link events and their services.
- **INNER JOIN Services AS S ON SE.Service_ID = S.Services_ID**
  Connects to the Services table to get the name of each service.
- **INNER JOIN Locations AS L ON E.Location_ID = L.Location_ID**
  Adds location details by joining with the Locations table.
- **INNER JOIN**
  Ensures only records with matching entries in all tables are included in the results.

Insights :

- **Comprehensive Event Overview**
  This query consolidates event, service, and location information, providing a unified view that supports efficient compliance tracking and operational oversight.
- **Informed Resource Coordination**
  By linking services to specific events and locations, businesses can better plan logistics and allocate resources where they are needed most.

1a. INNER Join Statement Query :

```
SELECT
    A.Address_ID,
    A.Address_Details,
    E.Event_Outcome_Description,
    S.Service_Type_Description
FROM
    Addresses A

INNER JOIN
    Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code

INNER JOIN
     Ref_service_Types S ON A.Address_ID = S.Service_Type_Code;
```

Results :

Matching Records Across Tables

Results Messages

	Address_ID	Address_Details	Event_Outcome_Description	Service_Type_Description
1	1	123 Main St	Invalid input provided	Workplace Hazard Assessment
2	2	616 Little Fleur Place	Invalid input provided	Emergency Response Planning
3	3	7 Manley Way	Partial success	First Aid Training
4	4	06750 Delaware Junction	Closed	Safety Training
5	5	9 Jackson Terrace	Postponed	Safety Training
6	6	02429 Carey Street	Invalid input provided	Vaccination Drive
7	7	5643 Melody Center	Connection timeout	Fire Safety Inspection
8	8	5267 Debra Terrace	Permission denied	Vaccination Drive
9	9	47738 Spenser Circle	System malfunction	Health Checkup
10	10	61492 Lakeland Point	Follow-up Required	First Aid Training
11	11	2319 Londonderry Hill	Invalid input provided	Emergency Response Planning
12	12	33293 Kingsford Avenue	In Progress	Emergency Response Planning
13	13	063 Westport Parkway	Partial success	Emergency Response Planning
14	14	52779 Northwestern Drive	Permission denied	Health Checkup
15	15	33171 Mayer Crossing	System malfunction	Incident Investigation
16	16	5888 Pearson Point	Successful completion	Fire Safety Inspection

Explanation :

- SELECT ...
  Retrieves the address ID, address details, event outcome description, and service type description for each matching record.

- **FROM Addresses A**
  Uses the Addresses table as the starting point, with the alias A.
- **INNER JOIN Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code**
  Joins with Ref_Event_Outcomes to match addresses with their corresponding event outcomes.
- **INNER JOIN Ref_service_Types S ON A.Address_ID = S.Service_Type_Code**
  Joins with Ref_service_Types to match addresses with their corresponding service types.
- **INNER JOIN**
  Includes only records where the address ID matches codes in both referenced tables, ensuring all data is related.

Insights :

- **Location-Based Analysis**
  This query pinpoints which addresses are linked to specific event outcomes and service types, making it easier to analyze patterns or results at particular locations.
- **Service and Outcome Tracking**
  By connecting services and outcomes to addresses, the query helps organizations evaluate service delivery success and address-specific issues more effectively.

2.

Comprehensive Event Data Analysis

" As a Business Analyst, I want to use a full outer join to retrieve all event details, including unmatched records, so that I can analyze gaps and ensure no data is overlooked."

2. Full Outer Join Statement Query :

```
SELECT E.Event_ID AS "Event ID",
       E.Start_Date AS "Start Date",
       E.End_Date AS "End Date",
       S.Service_Name AS "Service Name"
FROM Events AS E
FULL OUTER JOIN Services_In_Events AS SE ON E.Event_ID = SE.Event_ID
FULL OUTER JOIN Services AS S ON SE.Service_ID = S.Services_ID;
```

Results :

Results Messages

	Event ID	Start Date	End Date	Service Name
1	1	2024-07-26	2025-01-01	Personal Training
2	2	2024-04-18	2025-01-01	Event Planning
3	3	2024-08-22	2025-01-01	Event Planning
4	4	2025-01-11	2025-01-01	Event Planning
5	5	2024-12-23	2025-01-01	Landscaping
6	6	2025-03-14	2025-01-01	Pet Grooming
7	7	2024-05-11	2025-01-01	Home Renovation
8	8	2024-07-11	2025-01-01	Home Renovation
9	9	2025-01-20	2025-01-01	Pet Grooming
10	10	2024-05-03	2025-01-01	Massage Therapy

Explanation :

- **SELECT ...**
  Retrieves event IDs, dates, and service names for analysis.
- **FROM Events AS E**
  Starts with the Events table.
- **FULL OUTER JOIN Services_In_Events AS SE ON E.Event_ID = SE.Event_ID**
  Includes all events and their service links, even unmatched ones.
- **FULL OUTER JOIN Services AS S ON SE.Service_ID = S.Services_ID**
  Ensures all services appear, linked or not.

Insights :

- **Gap Identification**
  Highlights events without services and services without events, helping to find missing connections.
- **Comprehensive Planning**
  Provides a full dataset to support better resource allocation and decision-making.

2a.

FULL OUTER JOIN Statement :

"As a Data Analyst, I want to use a Full OUTER JOIN query across Addresses, Ref_Event_Outcomes, and Ref_Service_Types tables to retrieve all records, including unmatched ones, so that I can identify any data gaps or inconsistencies in address-related event outcomes and service types."

Query :

```
SELECT
    COALESCE(A.Address_ID, E.Event_Outcome_Code,
    S.Service_Type_Code) AS ID,
    A.Address_Details,
    E.Event_Outcome_Description,
    S.Service_Type_Description
FROM
    Addresses A
FULL OUTER JOIN
   Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code
FULL OUTER JOIN
    Ref_Service_Types S ON A.Address_ID = S.Service_Type_Code;
```

Results :

Results Messages

	ID	Address_Details	Event_Outcome_Description	Service_Type_Description
1	1	123 Main St	Invalid input provided	Workplace Hazard Assessment
2	2	616 Little Fleur Place	Invalid input provided	Emergency Response Planning
3	3	7 Manley Way	Partial success	First Aid Training
4	4	06750 Delaware Junction	Closed	Safety Training
5	5	9 Jackson Terrace	Postponed	Safety Training
6	6	02429 Carey Street	Invalid input provided	Vaccination Drive
7	7	5643 Melody Center	Connection timeout	Fire Safety Inspection
8	8	5267 Debra Terrace	Permission denied	Vaccination Drive
9	9	47738 Spenser Circle	System malfunction	Health Checkup
10	10	61492 Lakeland Point	Follow-up Required	First Aid Training
11	11	2319 Londonderry Hill	Invalid input provided	Emergency Response Planning

Explanation :

- **SELECT COALESCE(A.Address_ID, E.Event_Outcome_Code, S.Service_Type_Code) AS ID, ...** Uses COALESCE to display the first non-null ID from any of the three tables, ensuring every row has an identifier.
- **FROM Addresses A**
  Begins with the Addresses table as the primary source.
- **FULL OUTER JOIN Ref_Service_Types S ON A.Address_ID**
  Further combines all records from the Services table, preserving unmatched entries as well.
- **FULL OUTER JOIN Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code**
  Merges all records from both tables, even if there is no match.

Insights :

- **Data Completeness**
  Offers a unified view of all addresses, event outcomes, and service types, including unmatched records for thorough analysis.
- **Error Detection**
  Helps identify missing links or mismatches between addresses, event outcomes, and service types for improved data quality.

3.

LEFT JOIN Statement :

"As a Business Analyst, I want to retrieve event details along with their outcomes using a LEFT JOIN query so that I can analyze all events, including those without defined outcomes."

Query :

```
SELECT E.Event_ID AS "Event ID",
       E.Start_Date AS "Start Date",
       E.End_Date AS "End Date",
       R.Event_Outcome_Description AS "Outcome"
FROM Events AS E
       LEFT JOIN Ref_Event_Outcomes AS R ON E.Event_Outcome_Code
       = R.Event_Outcome_Code;
```

Results :

Results Messages

	Event ID	Start Date	End Date	Outcome
1	1	2024-07-26	2025-01-01	Partial success
2	2	2024-04-18	2025-01-01	Partial success
3	3	2024-08-22	2025-01-01	Invalid input provided
4	4	2025-01-11	2025-01-01	Successful completion
5	5	2024-12-23	2025-01-01	Partial success
6	6	2025-03-14	2025-01-01	Partial success
7	7	2024-05-11	2025-01-01	System malfunction
8	8	2024-07-11	2025-01-01	Invalid input provided
9	9	2025-01-20	2025-01-01	Invalid input provided
10	10	2024-05-03	2025-01-01	Permission denied

Explanation :

- **SELECT ...**
  Retrieves event IDs, start and end dates, and their outcome descriptions.
- **FROM Events AS E**
  Uses the Events table as the main source.
- **LEFT JOIN Ref_Event_Outcomes AS R ON E.Event_Outcome_Code = R.Event_Outcome_Code**
  Includes all events, matching them with outcomes where available, and keeps events in the results even if no outcome is found.

Insights :

- **Outcome Completeness**
  Highlights events that lack defined outcomes, making it easier to spot where follow-up or data updates are needed.
- **Performance Clarity**
  Provides a full overview of event results, supporting more informed operational decisions and improvements.

3a.

LEFT JOIN Statement :

"As a Data Analyst, I want to view all addresses with any related event outcomes and service types, so I can identify addresses even if some details are missing."

Query :

```
SELECT
    A.Address_ID,
    A.Address_Details,
    E.Event_Outcome_Description,
      S.Service_Type_Description
  FROM
      Addresses A
  LEFT JOIN
      Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code
  LEFT JOIN
      Ref_service_Types S ON A.Address_ID = S.Service_Type_Code;
```

Results :

Results Messages

	Address_ID	Address_Details	Event_Outcome_Description	Service_Type_Description
1	1	123 Main St	Invalid input provided	Workplace Hazard Assessment
2	2	616 Little Fleur Place	Invalid input provided	Emergency Response Planning
3	3	7 Manley Way	Partial success	First Aid Training
4	4	06750 Delaware Junction	Closed	Safety Training
5	5	9 Jackson Terrace	Postponed	Safety Training
6	6	02429 Carey Street	Invalid input provided	Vaccination Drive
7	7	5643 Melody Center	Connection timeout	Fire Safety Inspection
8	8	5267 Debra Terrace	Permission denied	Vaccination Drive
9	9	47738 Spenser Circle	System malfunction	Health Checkup
10	10	61492 Lakeland Point	Follow-up Required	First Aid Training
11	11	2319 Londonderry Hill	Invalid input provided	Emergency Response Planning
12	12	33293 Kingsford Avenue	In Progress	Emergency Response Planning
13	13	063 Westport Parkway	Partial success	Emergency Response Planning
14	14	52779 Northwestern Drive	Permission denied	Health Checkup
15	15	33171 Mayer Crossing	System malfunction	Incident Investigation
16	16	5888 Pearson Point	Successful completion	Fire Safety Inspection

Explanation :

- **SELECT ...**
  Retrieves address IDs, address details, event outcome descriptions, and service type descriptions.
- **FROM Addresses A**
  Starts with the Addresses table as the primary source.
- **LEFT JOIN Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code**
  Includes all addresses and matches event outcomes where available, returning NULL if there's no
  match.
- **LEFT JOIN Ref_service_Types S ON A.Address_ID = S.Service_Type_Code**
  Matches service types to addresses, also returning NULL for unmatched records.

Insights :

- **Coverage Gaps**

  Highlights addresses without linked event outcomes or services, making it easier to spot areas needing attention.
- **Data Completeness** Helps identify missing or incomplete data, supporting efforts to improve service coverage and database accuracy.

4.
RIGHT JOIN Statement :

"As an Operations Manager, I want to link services with their corresponding events to ensure all services are accounted for and properly scheduled."

Query :

```
SELECT          S.Service_Name AS "Service Name",
                E.Event_ID AS "Event ID",
                E.Start_Date AS "Start Date"
FROM            Services_In_Events AS SE
   RIGHT JOIN
              Events AS E ON SE.Event_ID = E.Event_ID
   RIGHT JOIN
              Services AS S ON SE.Service_ID = S.Services_ID;
```

Results :

Results Messages

	Service Name	Event ID	Start Date
1	Personal Training	1	2024-07-26
2	Event Planning	2	2024-04-18
3	Event Planning	3	2024-08-22
4	Event Planning	4	2025-01-11
5	Landscaping	5	2024-12-23
6	Pet Grooming	6	2025-03-14
7	Home Renovation	7	2024-05-11
8	Home Renovation	8	2024-07-11
9	Pet Grooming	9	2025-01-20
10	Massage Therapy	10	2024-05-03

Explanation :

- **SELECT ...**
  Retrieves service names, event IDs, and start dates for analysis.
- **FROM Services_In_Events AS SE**
  Begins with the Services_In_Events table as the base.
- **RIGHT JOIN Events AS E ON SE.Event_ID = E.Event_ID**
  Ensures all events are included, even if they have no linked services.
- **RIGHT JOIN Services AS S ON SE.Service_ID = S.Services_ID**
  Ensures all services are included, even if not linked to any event.

Insights :

- **Utilization Gaps**
  Identifies services that are not currently associated with events, helping to uncover underutilized resources.
  Association Overview
- Provides a detailed mapping of which services are linked to which events, supporting better planning and operational efficiency.

4a.
RIGHT JOIN Statement :

"As a Healthcare Administrator, I want to retrieve Event_Outcomes, their associated addresses, and the Service_Types offered, so I can evaluate service coverage and resource allocation across different locations."

Query :

```
SELECT
    E.Event_Outcome_Code,
    E.Event_Outcome_Description,
    A.Address_Details,
    S.Service_Type_Description
FROM
   Ref_Event_Outcomes E
RIGHT JOIN
    Addresses A ON E.Event_Outcome_Code = A.Address_ID
RIGHT JOIN
    Ref_Service_Types S ON S.Service_Type_Code = A.Address_ID;
```

Results :

Results Messages

	Event_Outcome_Code	Event_Outcome_Description	Address_Details	Service_Type_Description
1	1	Invalid input provided	123 Main St	Workplace Hazard Assessment
2	2	Invalid input provided	616 Little Fleur Place	Emergency Response Planning
3	3	Partial success	7 Manley Way	First Aid Training
4	4	Closed	06750 Delaware Junction	Safety Training
5	5	Postponed	9 Jackson Terrace	Safety Training
6	6	Invalid input provided	02429 Carey Street	Vaccination Drive
7	7	Connection timeout	5643 Melody Center	Fire Safety Inspection
8	8	Permission denied	5267 Debra Terrace	Vaccination Drive
9	9	System malfunction	47738 Spenser Circle	Health Checkup
10	10	Follow-up Required	61492 Lakeland Point	First Aid Training
11	11	Invalid input provided	2319 Londonderry Hill	Emergency Response Planning
12	12	In Progress	33293 Kingsford Avenue	Emergency Response Planning
13	13	Partial success	063 Westport Parkway	Emergency Response Planning
14	14	Permission denied	52779 Northwestern Drive	Health Checkup
15	15	System malfunction	33171 Mayer Crossing	Incident Investigation

Explanation :

- **SELECT ...**
  Retrieves event outcome codes and descriptions, address details, and service type descriptions.
- **FROM Ref_Event_Outcomes E**
  Starts with the reference table for event outcomes.
- **RIGHT JOIN Addresses A ON E.Event_Outcome_Code = A.Address_ID**
  Ensures all addresses are included, even if there's no matching event outcome.
- **RIGHT JOIN Ref_Service_Types S ON S.Service_Type_Code = A.Address_ID**
  Ensures all service types are included, even if not linked to an address.

Insights :

- **Outcome Accountability**
  Ensures every event outcome is matched with a location and service, highlighting any gaps in event resolution.
  Unresolved Tracking
- Helps identify events that may not yet be addressed, supporting follow-up and improved case management.

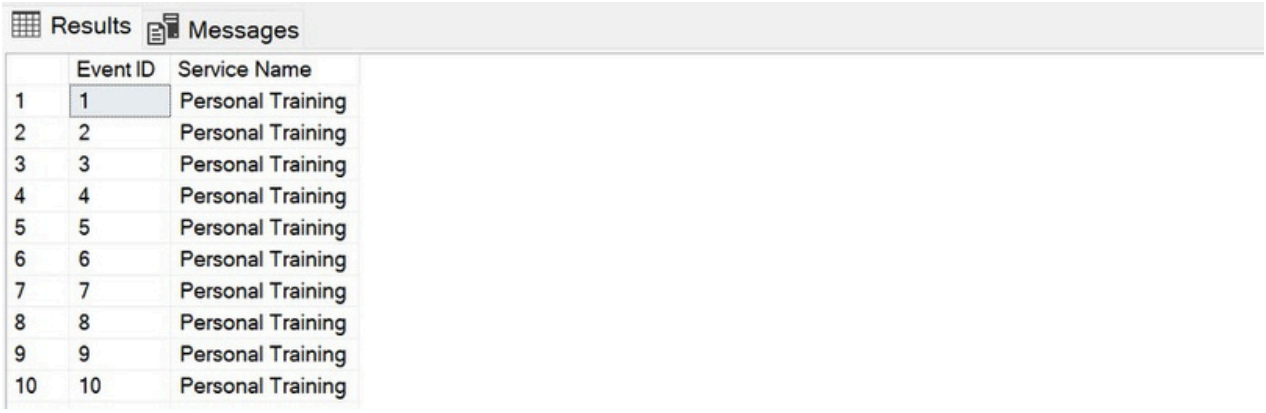
5.
CROSS JOIN Statement

"As a Database Analyst, I want to generate all possible combinations of events and services so that I can explore potential pairings."

Query :

```
SELECT

    E.Event_ID AS "Event ID",

    S.Service_Name AS "Service Name"

FROM

    Events AS E

CROSS JOIN

    Services AS S;
```

Results :



Results | Messages

	Event ID	Service Name
1	1	Personal Training
2	2	Personal Training
3	3	Personal Training
4	4	Personal Training
5	5	Personal Training
6	6	Personal Training
7	7	Personal Training
8	8	Personal Training
9	9	Personal Training
10	10	Personal Training

Explanation :

- **SELECT ...**
  Retrieves event IDs and service names for each combination.
- **FROM Events AS E**
  Uses the Events table as one source.
- **CROSS JOIN Services AS S**
  Combines every event with every service, generating all possible pairs.

Insights :

- **Opportunity Exploration**
  Provides a complete list of all possible event-service pairings, helping to identify untapped service opportunities.
  Innovation Potential
- Enables businesses to consider new and creative event-service combinations for developing unique offerings.

5a.

CROSS JOIN Statement

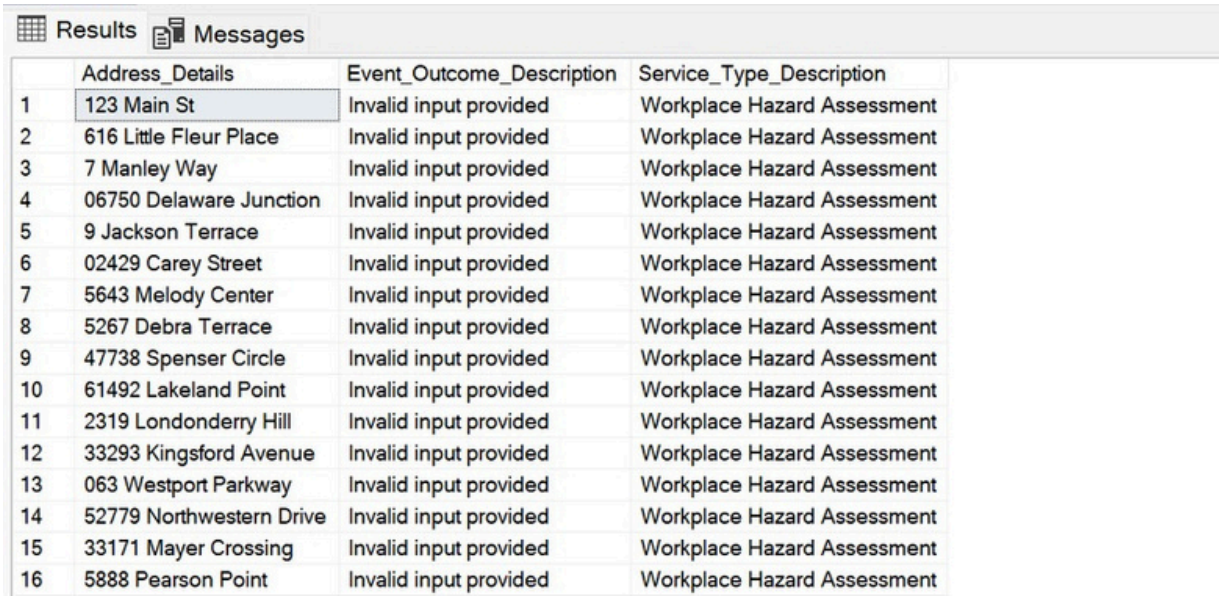
"As a Data Analyst, I want to generate all possible combinations of addresses, event outcomes, and service types, so that I can explore potential relationships and uncover unexpected patterns."

Query :

```
SELECT
        A.Address_Details,

        E.Event_Outcome_Description,

        S.Service_Type_Description
```

```
FROM
            Addresses A
CROSS JOIN
            Ref_Event_Outcomes E
CROSS JOIN
           Ref_service_Types S;
```

Results :



Results | Messages

	Address_Details	Event_Outcome_Description	Service_Type_Description
1	123 Main St	Invalid input provided	Workplace Hazard Assessment
2	616 Little Fleur Place	Invalid input provided	Workplace Hazard Assessment
3	7 Manley Way	Invalid input provided	Workplace Hazard Assessment
4	06750 Delaware Junction	Invalid input provided	Workplace Hazard Assessment
5	9 Jackson Terrace	Invalid input provided	Workplace Hazard Assessment
6	02429 Carey Street	Invalid input provided	Workplace Hazard Assessment
7	5643 Melody Center	Invalid input provided	Workplace Hazard Assessment
8	5267 Debra Terrace	Invalid input provided	Workplace Hazard Assessment
9	47738 Spenser Circle	Invalid input provided	Workplace Hazard Assessment
10	61492 Lakeland Point	Invalid input provided	Workplace Hazard Assessment
11	2319 Londonderry Hill	Invalid input provided	Workplace Hazard Assessment
12	33293 Kingsford Avenue	Invalid input provided	Workplace Hazard Assessment
13	063 Westport Parkway	Invalid input provided	Workplace Hazard Assessment
14	52779 Northwestern Drive	Invalid input provided	Workplace Hazard Assessment
15	33171 Mayer Crossing	Invalid input provided	Workplace Hazard Assessment
16	5888 Pearson Point	Invalid input provided	Workplace Hazard Assessment

Explanation :

- SELECT ... Retrieves address details, event outcome descriptions, and service type descriptions for each combination.
- **FROM Addresses A**
  Uses the Addresses table as one source.
- **CROSS JOIN Ref_Event_Outcomes E**
  Combines every address with every event outcome.
- **CROSS JOIN Ref_service_Types S**
  Further combines each address-event outcome pair with every service type, producing all possible combinations.

Insights :

- **Scenario Simulation**
  Generates every possible combination of addresses, event outcomes, and service types, which is valuable for testing and scenario analysis.
- **Relationship Discovery**
  Helps uncover potential patterns or relationships between data points that might not be directly linked in the actual data.

Demo B2: How to write Query with Subqueries

1.

Events with Specific Services

" As a Health and Safety Officer, I want to identify events that involve specific services so that I can ensure compliance with regulations. "

Subquery in the WHERE Clause :

```
SELECT Event_ID, Start_Date, End_Date
FROM Events
WHERE Event_ID IN ( SELECT Event_ID FROM Services_In_Events
      WHERE Service_ID5 );
```

Results :

Results Messages

	Event_ID	Start_Date	End_Date
1	5	2024-12-23	2025-01-01

Explanation :

- **SELECT Event_ID, Start_Date, End_Date**
  Retrieves the event ID, start date, and end date for matching events.
- **FROM Events**
  Uses the Events table as the main data source.
- **WHERE Event_ID IN (SELECT Event_ID FROM Services_In_Events WHERE Service_ID = 5)**
  Filters events to include only those whose IDs appear in the Services_In_Events table with the specified Service_ID (in this case, 5).

Insights :

- **Compliance Assurance**
  Helps Health and Safety Officers quickly find events involving specific services, supporting targeted compliance checks and regulatory adherence.
- **Targeted Management**
  Enables efficient tracking and management of events by service type, improving resource allocation and safety planning.

1a.

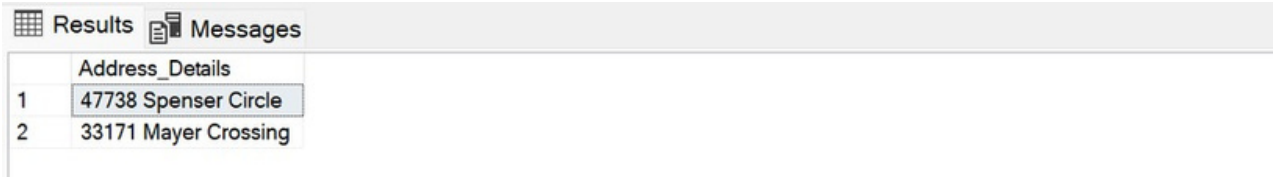
Addresses with System Malfunction Event Outcomes

"As a Health and Safety Officer, I want to filter records based on conditions derived from related tables so that I can focus on problem areas and take corrective actions effectively."

Subquery in the WHERE Clause :

```
SELECT  Address_Details
FROM Addresses
WHERE Address_ID IN (SELECT Event_Outcome_Code FROM
      Ref_Event_Outcomes
WHERE Event_Outcome_Description = 'System malfunction');
```

Results :


Results Messages

	Address_Details
1	47738 Spenser Circle
2	33171 Mayer Crossing

Explanation :

- **SELECT Address_Details**
  Retrieves address details for relevant locations.
- **FROM Addresses**
  Uses the Addresses table as the main source.
- **WHERE Address_ID IN (SELECT Event_Outcome_Code FROM Ref_Event_Outcomes WHERE Event_Outcome_Description = 'System malfunction')**
  Filters addresses to only those whose IDs match event outcome codes associated with a 'System malfunction', using a subquery in the WHERE clause.

Insights :

- **Problem Area Identification**
  Helps pinpoint specific locations where system malfunctions have occurred, enabling focused attention on trouble spots.
- **Maintenance Prioritization**
  Supports efficient troubleshooting and resource allocation by highlighting addresses that require urgent maintenance or corrective action.

2.

Counting Services in Events

"As a Service Manager, I want to count the number of services provided during each event so that I can analyze service utilization and optimize resource planning."

Subquery in the SELECT Clause Query :

```
SELECT Event_ID, Start_Date, End_Date,
(SELECT COUNT(Service_ID)
 FROM Services_In_Events
 WHERE Services_In_Events.Event_ID =
       Events.Event_ID) AS Total_Services
FROM Events;
```

Results :

Results | Messages

	Event_ID	Start_Date	End_Date	Total_Services
1	1	2024-07-26	2025-01-01	1
2	2	2024-04-18	2025-01-01	1
3	3	2024-08-22	2025-01-01	1
4	4	2025-01-11	2025-01-01	1
5	5	2024-12-23	2025-01-01	1
6	6	2025-03-14	2025-01-01	1
7	7	2024-05-11	2025-01-01	1
8	8	2024-07-11	2025-01-01	1
9	9	2025-01-20	2025-01-01	1
10	10	2024-05-03	2025-01-01	1

Explanation:

- **SELECT Event_ID, Start_Date, End_Date**
  Retrieves the unique event identifier along with its start and end dates from the Events table.
- **(SELECT COUNT(Service_ID) FROM Services_In_Events WHERE Services_In_Events.Event_ID = Events.Event_ID) AS Total_Services**
  Uses a correlated subquery to count how many services are linked to each event by matching Event_IDs.
- **FROM Events**
  Specifies the Events table as the main data source for the query.

Insights:

- **Resource Optimization**
  Enables more effective planning and allocation of resources by revealing the service load associated with each event.
- **Service Utilization Analysis** Shows the number of services provided for each event, helping managers understand which events require more support.

2a.

Enrich Address Data with Service Details Using Subquery

As a Data Analyst, I want to enrich data by adding information from related tables directly into the result set and combine location along with service details for operational reporting so that I can generate comprehensive reports that provide insights into service delivery and resource management.

Subquery in the SELECT Clause Query :

```
SELECT A.Address_Details,
       (SELECT Service_Type_Description FROM Ref_service_Types
       S WHERE S.Service_Type_Code = A.Address_ID)
       AS Service_Description
FROM   Addresses A;
```

Results :

Results Messages

	Address_Details	Service_Description
1	123 Main St	Workplace Hazard Assessment
2	616 Little Fleur Place	Emergency Response Planning
3	7 Manley Way	First Aid Training
4	06750 Delaware Junction	Safety Training
5	9 Jackson Terrace	Safety Training
6	02429 Carey Street	Vaccination Drive
7	5643 Melody Center	Fire Safety Inspection
8	5267 Debra Terrace	Vaccination Drive
9	47738 Spenser Circle	Health Checkup
10	61492 Lakeland Point	First Aid Training
11	2319 Londonderry Hill	Emergency Response Planning
12	33293 Kingsford Avenue	Emergency Response Planning
13	063 Westport Parkway	Emergency Response Planning
14	52779 Northwestern Drive	Health Checkup
15	33171 Mayer Crossing	Incident Investigation

Explanation :

- **SELECT A.Address_Details**
  Retrieves address details (e.g., location names or addresses) from the **Addresses** table.
  (SELECT Service_Type_Description FROM Ref_service_Types S WHERE
- **S.Service_Type_Code = A.Address_ID) AS Service_Description**
  Uses a **correlated subquery** to fetch the service type description from **Ref_service_Types** where the service code matches the address ID.
- **FROM Addresses A**
  Specifies the **Addresses** table as the primary data source.

Insights :

- **Service-Location Alignment**
  Verifies which services are linked to specific addresses, ensuring service offerings match their intended locations (e.g., confirming "First Aid Station" is mapped to a clinic address).
- **Operational Reporting**
  Combines location and service details in one output, streamlining reports for resource management and service delivery tracking.

3.

Filter Events by Service Type

"As a Health and Safety Officer, I want to identify events that have a specific service associated with them so that I can ensure compliance with safety regulations for targeted service types."

Correlated Subquery :

```
SELECT Event_ID,Start_Date,End_Date
FROM Events AS E1
WHERE EXISTS ( SELECT 1
FROM Services_In_Events AS SE
WHERE SE.Event_ID = E1.Event_ID AND SE.Service_ID = 5);
```

Results :

Results Messages

	Event_ID	Start_Date	End_Date
1	5	2024-12-23	2025-01-01

Explanation :

- **SELECT Event_ID, Start_Date, End_Date**
  Retrieves the unique identifier and date range for each event.
- **FROM Events AS E1**
  Uses the Events table, assigning it the alias E1 for reference in the subquery.
- **WHERE EXISTS (SELECT 1 FROM Services_In_Events AS SE WHERE SE.Event_ID = E1.Event_ID AND SE.Service_ID = 5)**
  Applies a correlated subquery that checks if an event is associated with the specific service (Service_ID = 5), efficiently filtering only those events.

Insights :

- **Efficient Filtering**
  The correlated subquery with EXISTS quickly identifies relevant events by stopping at the first match, making it more efficient than joins for this scenario.
- **Targeted Compliance Monitoring**
  By isolating events linked to a specific regulated service, the query enables focused compliance checks and prioritization of safety-related inspections.

3a.

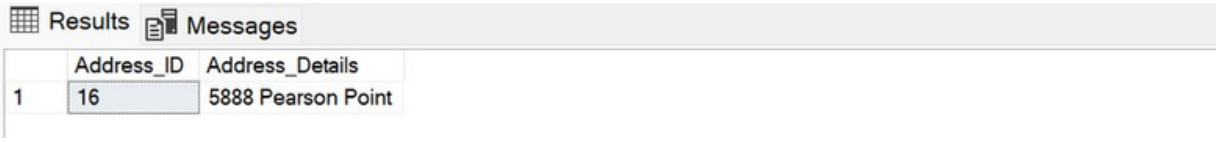
Addresses with Successful Event Outcomes

"As a Community Outreach Coordinator, I want to identify addresses where events have resulted in 'Successful completion' outcomes so that I can focus resources on replicating successful strategies in similar areas."

Correlated Subquery :

```
SELECT A.Address_ID,
       A.Address_Details
FROM   Addresses A
WHERE
    EXISTS (SELECT 1 FROM Ref_Event_Outcomes E
            WHERE E.Event_Outcome_Code = A.Address_ID AND
            E.Event_Outcome_Description
    = 'Successful completion');
```

Results :



Results Messages

	Address_ID	Address_Details
1	16	5888 Pearson Point

Explanation :

- **SELECT A.Address_ID, A.Address_Details**
  Retrieves the unique address identifier and its details from the Addresses table.
- **FROM Addresses A**
  Uses the Addresses table as the main data source, assigning it the alias A.
- **WHERE EXISTS (SELECT 1 FROM Ref_Event_Outcomes E WHERE E.Event_Outcome_Code = A.Address_ID AND E.Event_Outcome_Description = 'Successful completion')**
  Uses a correlated subquery to check for each address if there is at least one event outcome labeled 'Successful completion' linked to it.

Insights :

- **Success Location Identification**
  Pinpoints addresses where events have achieved successful outcomes, helping to recognize areas of effective community engagement.
- **Strategy Replication**
  Enables the organization to focus resources on locations with proven success, supporting the replication of effective outreach strategies in similar areas.

4.

Events Exceeding Average Duration

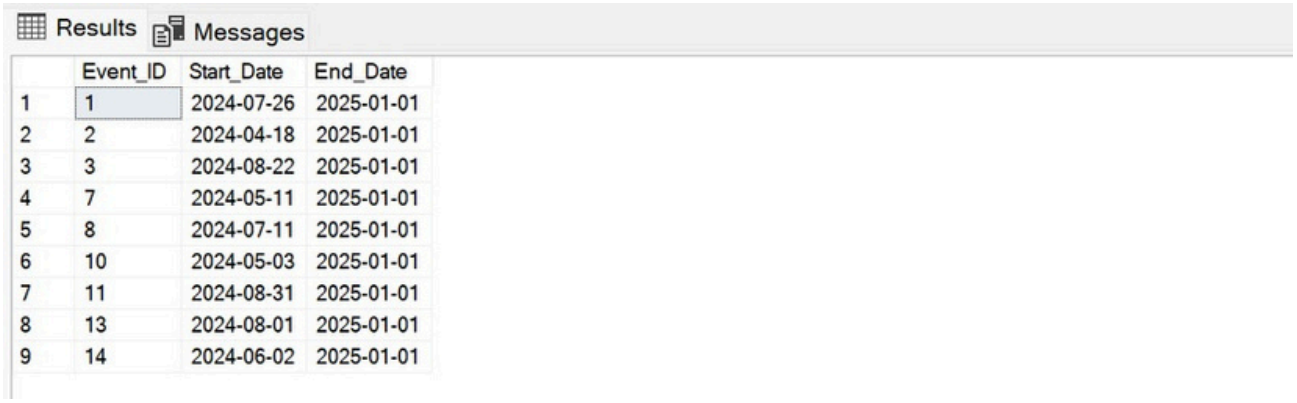
"As an Event Analyst, I want to identify events whose duration (in days) exceeds the average duration of all events so that I can focus on analyzing and understanding factors contributing to longer events."

Non-Correlated Subquery :

```
SELECT Event_ID,

       Start_Date,

       End_Date
```

```
FROM    Events

WHERE   DATEDIFF( DAY, Start_Date, End_Date) >

(SELECT AVG( DATEDIFF(DAY, Start_Date, End_Date)) FROM Events );
```

Results :



Results | Messages

	Event_ID	Start_Date	End_Date
1	1	2024-07-26	2025-01-01
2	2	2024-04-18	2025-01-01
3	3	2024-08-22	2025-01-01
4	7	2024-05-11	2025-01-01
5	8	2024-07-11	2025-01-01
6	10	2024-05-03	2025-01-01
7	11	2024-08-31	2025-01-01
8	13	2024-08-01	2025-01-01
9	14	2024-06-02	2025-01-01

Explanation :

- **SELECT Event_ID, Start_Date, End_Date**
  Retrieves the unique identifier and date range for each event from the Events table.
- **FROM Events**
  Specifies the Events table as the source of event data.
- **WHERE DATEDIFF(DAY, Start_Date, End_Date) > (SELECT AVG(DATEDIFF(DAY, Start_Date, End_Date)) FROM Events)**
  Uses a non-correlated subquery to calculate the average event duration and filters for events whose duration exceeds this average.

Insights :

- **Identification of Extended Events**
  Highlights events that last longer than the typical duration, enabling focused analysis on outliers.
- **Root Cause Analysis**
  Supports investigation into the reasons behind longer event durations, which can inform improvements in planning, logistics, and resource allocation.

4a.

Event Outcome Count by Address

"As a Performance Analyst, I want to aggregate data to summarize information for reporting purposes and measure performance or activity levels at specific locations by counting the number of event outcomes associated with each address so that I can evaluate the effectiveness of events at different locations."

Non-Correlated Subquery :

```
SELECT Address_Details,
        (SELECT COUNT(*) FROM Ref_Event_Outcomes WHERE
        Ref_Event_Outcomes.Event_Outcome_Code =
        Addresses.Address_ID)
        AS Event_Count
FROM   Addresses;
```

Results :

Results | Messages

	Address_Details	Event_Count
1	123 Main St	1
2	616 Little Fleur Place	1
3	7 Manley Way	1
4	06750 Delaware Junction	1
5	9 Jackson Terrace	1
6	02429 Carey Street	1
7	5643 Melody Center	1
8	5267 Debra Terrace	1
9	47738 Spenser Circle	1
10	61492 Lakeland Point	1
11	2319 Londonderry Hill	1
12	33293 Kingsford Avenue	1
13	063 Westport Parkway	1
14	52779 Northwestern Drive	1
15	33171 Mayer Crossing	1

Explanation :

- **SELECT Address_Details**
  Retrieves the descriptive details for each address from the Addresses table.
- **(SELECT COUNT(*) FROM Ref_Event_Outcomes WHERE Ref_Event_Outcomes.Event_Outcome_Code = Addresses.Address_ID) AS Event_Count**
  Uses a non-correlated subquery to count how many event outcomes are associated with each address, returning the count as a new column.
- **FROM Addresses**
  Specifies the Addresses table as the main data source.

Insights :

- **Activity Measurement**
  Shows the number of event outcomes linked to each address, providing a clear measure of activity or engagement at specific locations.
- **Resource Allocation**
  Helps identify which locations are most or least active, supporting data-driven decisions for allocating resources and planning future events.

5.

Average Event Duration Calculation

"As a Data Analyst, I want to calculate the average duration of all events so that I can benchmark event timelines and identify trends in event length over time."

Subquery in the FROM Clause :

```
SELECT AVG(Event_Duration) AS Average_Duration FROM (SELECT
        Event_ID,DATEDIFF(DAY, Start_Date, End_Date)
AS      Event_Duration FROM Events) AS DurationTable;
```

Results :

Explanation :

- **SELECT AVG(Event_Duration) AS Average_Duration**
  Calculates the average event duration across all events and returns it as a single summary value.
- **FROM (SELECT Event_ID, DATEDIFF(DAY, Start_Date, End_Date) AS Event_Duration FROM Events) AS DurationTable**
  The subquery computes the duration in days for each event using DATEDIFF and creates a temporary table (DurationTable) for the outer query to aggregate.

Insights :

- **Overall Trend Analysis**
  Provides a clear benchmark for average event duration, enabling analysts to monitor changes and trends over time.
- **Strategic Planning**
  Supports informed resource allocation and scheduling by understanding the typical length of events.

5a.

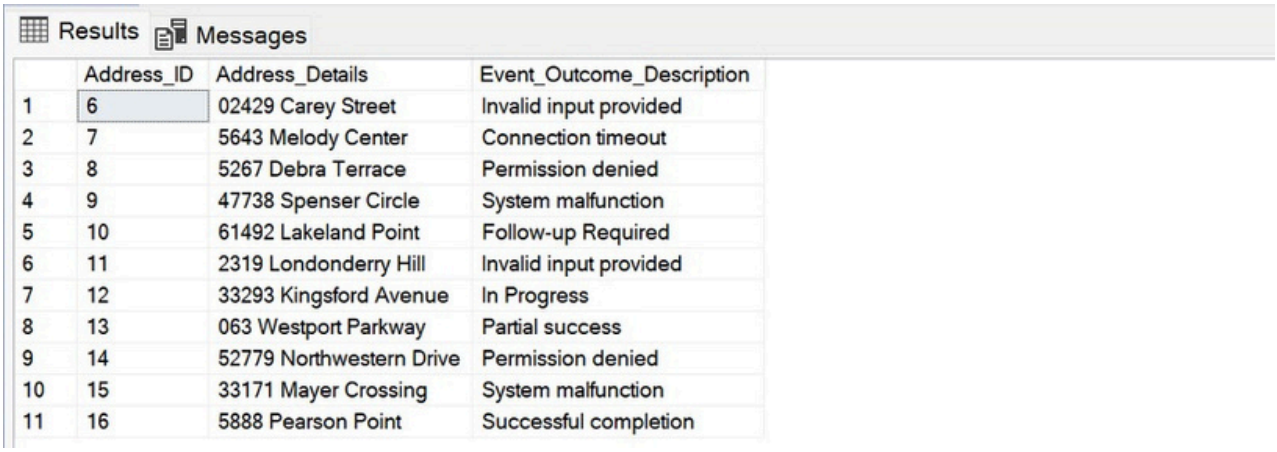
Filtered Address Analysis with Event Outcomes

"As a Resource Planner, I want to filter addresses based on an initial criterion, such as Address_ID being greater than 5, so that I can then analyze these specific locations in conjunction with their event outcomes to resource allocation and avoid scheduling conflicts. "

Subquery in the FROM Clause :

```
SELECT
    Sub.Address_ID,
    Sub.Address_Details,
    E.Event_Outcome_Description
FROM
    (SELECT Address_ID, Address_Details FROM Addresses
    WHERE Address_ID > 5) AS Sub
INNER JOIN
    Ref_Event_Outcomes E ON Sub.Address_ID = E.Event_Outcome_Code;
```

Results :



Results Messages

	Address_ID	Address_Details	Event_Outcome_Description
1	6	02429 Carey Street	Invalid input provided
2	7	5643 Melody Center	Connection timeout
3	8	5267 Debra Terrace	Permission denied
4	9	47738 Spenser Circle	System malfunction
5	10	61492 Lakeland Point	Follow-up Required
6	11	2319 Londonderry Hill	Invalid input provided
7	12	33293 Kingsford Avenue	In Progress
8	13	063 Westport Parkway	Partial success
9	14	52779 Northwestern Drive	Permission denied
10	15	33171 Mayer Crossing	System malfunction
11	16	5888 Pearson Point	Successful completion

Explanation :

- **SELECT Sub.Address_ID, Sub.Address_Details, E.Event_Outcome_Description**
  Retrieves the address ID, address details, and the description of the event outcome for each relevant record.
- **FROM (SELECT Address_ID, Address_Details FROM Addresses WHERE Address_ID > 5) AS Sub**
  Uses a subquery in the FROM clause to filter addresses with an ID greater than 5, creating a temporary table (Sub) for further analysis.
- **INNER JOIN Ref_Event_Outcomes E ON Sub.Address_ID = E.Event_Outcome_Code**
  Joins the filtered addresses with the Ref_Event_Outcomes table to associate each address with its corresponding event outcomes.

Insights :

- **Focused Location Analysis**
  Enables targeted review of specific addresses (those with Address_IDs greater than 5), which may represent higher-priority or recently added locations.
- **Enhanced Resource Planning**
  By linking filtered addresses with their event outcomes, the query supports more informed and strategic resource allocation for locations that meet the initial criterion.

Demo B3: How to Query with Self Joins

Finding Events with Overlapping Data's

" As an event coordinator, I want to identify events that overlap in their start and end dates to ensure proper resource allocation and avoid scheduling conflicts. "

Self Join Query :

```
SELECT  e1.Event_ID AS Event1,
        e2.Event_ID AS Event2,
        e1.Start_Date AS Event1_Start,
        e1.End_Date AS Event1_End,
        e2.Start_Date AS Event2_Start,
        e2.End_Date AS Event2_End
FROM    Events e1
        JOIN    Events e2
ON      e1.Event_ID != e2.Event_ID
        AND e1.Start_Date <= e2.End_Date
        AND e1.End_Date >= e2.Start_Date;
```

Results :

Results Messages

	Event1	Event2	Event1_Start	Event1_End	Event2_Start	Event2_End
1	2	1	2024-04-18	2025-01-01	2024-07-26	2025-01-01
2	3	1	2024-08-22	2025-01-01	2024-07-26	2025-01-01
3	5	1	2024-12-23	2025-01-01	2024-07-26	2025-01-01
4	7	1	2024-05-11	2025-01-01	2024-07-26	2025-01-01
5	8	1	2024-07-11	2025-01-01	2024-07-26	2025-01-01
6	10	1	2024-05-03	2025-01-01	2024-07-26	2025-01-01
7	11	1	2024-08-31	2025-01-01	2024-07-26	2025-01-01
8	12	1	2024-12-25	2025-01-01	2024-07-26	2025-01-01
9	13	1	2024-08-01	2025-01-01	2024-07-26	2025-01-01
10	14	1	2024-06-02	2025-01-01	2024-07-26	2025-01-01
11	15	1	2024-10-17	2025-01-01	2024-07-26	2025-01-01
12	16	1	2024-11-12	2025-01-01	2024-07-26	2025-01-01
13	1	2	2024-07-26	2025-01-01	2024-04-18	2025-01-01
14	3	2	2024-08-22	2025-01-01	2024-04-18	2025-01-01
15	5	2	2024-12-23	2025-01-01	2024-04-18	2025-01-01

Explanation :

- **SELECT e1.Event_ID AS Event1, e2.Event_ID AS Event2, e1.Start_Date AS Event1_Start, e1.End_Date AS Event1_End, e2.Start_Date AS Event2_Start, e2.End_Date AS Event2_End**
  Retrieves the IDs and date ranges for pairs of potentially overlapping events.
- **FROM Events e1 JOIN Events e2**
  Performs a self-join on the Events table to compare each event (e1) with every other event (e2).
- **ON e1.Event_ID != e2.Event_ID**
  Ensures an event is not compared with itself.
- **AND e1.Start_Date <= e2.End_Date AND e1.End_Date >= e2.Start_Date**
  Checks for overlapping date ranges between the two events.

Insights :

- **Self-Join for Comparison**
  The self-join enables the comparison of each event's date range with every other event, making it possible to detect overlaps.
- **Conflict and Resource Management**
  Identifying overlapping events helps in resolving scheduling conflicts and in planning shared resource allocation.

2.

Common Suffix Address Analysis

" As a Data Quality Analyst, I want to identify addresses that share the same last five characters in their details but have different IDs so that I can detect potential duplicates or related locations in the database."

Inner Join Query :

```
SELECT

    A1.Address_ID AS Address1_ID,

    A1.Address_Details AS Address1_Details,

    A2.Address_ID AS Address2_ID,

    A2.Address_Details AS Address2_Details
```

```
FROM
    Addresses A1
INNER JOIN
    Addresses A2 ON A1.Address_ID != A2.Address_ID
    AND RIGHT (A1.Address_Details, 5) =
    RIGHT(A2.Address_Details, 5);
```

Results :

Results Messages

	Address1_ID	Address1_Details	Address2_ID	Address2_Details
1	2	616 Little Fleur Place	19	90741 Golf Place
2	5	9 Jackson Terrace	8	5267 Debra Terrace
3	6	02429 Carey Street	20	442 Rieder Street
4	8	5267 Debra Terrace	5	9 Jackson Terrace
5	10	61492 Lakeland Point	16	5888 Pearson Point
6	16	5888 Pearson Point	10	61492 Lakeland Point
7	19	90741 Golf Place	2	616 Little Fleur Place
8	20	442 Rieder Street	6	02429 Carey Street

Explanation :

- **SELECT A1.Address_ID AS Address1_ID, A1.Address_Details AS Address1_Details, A2.Address_ID AS Address2_ID, A2.Address_Details AS Address2_Details**
  Retrieves pairs of address IDs and their details for addresses that meet the matching criteria.
- **FROM Addresses A1 INNER JOIN Addresses A2**
  Performs a self join on the Addresses table, allowing comparison between different rows within the same table.
- **ON A1.Address_ID != A2.Address_ID**
  Ensures that an address is not compared with itself.
- **AND RIGHT(A1.Address_Details, 5) = RIGHT(A2.Address_Details, 5)**
  Compares the last five characters of the Address_Details field to find addresses with matching suffixes.

Insights :

- **Duplicate and Related Address Detection**
  Helps uncover potential duplicate addresses or addresses that may be related due to similar suffixes, which is valuable for data deduplication and quality improvement.
- **Pattern-Based Grouping**
  Identifies addresses that might belong to the same street, building, or area, especially when address formatting is inconsistent.

c. Module 5: Sorting and Filtering Data

Why use Sorting and Filtering Data?

Sorting and filtering are essential SQL tools for targeted data extraction and analysis. They enable you to arrange and isolate specific subsets of data, leading to more informed and efficient decision-making.

Demo C1: Sorting with Order By Code

1.

Sorting Data by Codes for Enhanced Organization

" As a Training Coordinator, I want to list events sorted by their start date so that I can easily view the order in which they are scheduled. "

Order By Clause Query :

```
SELECT    Event_ID, Start_Date, End_Date

FROM      Events

ORDER BY  Start_Date ASC;
```

Results :

Results Messages

	Event_ID	Start_Date	End_Date
1	2	2024-04-18	2025-01-01
2	10	2024-05-03	2025-01-01
3	7	2024-05-11	2025-01-01
4	14	2024-06-02	2025-01-01
5	8	2024-07-11	2025-01-01
6	1	2024-07-26	2025-01-01
7	13	2024-08-01	2025-01-01
8	3	2024-08-22	2025-01-01
9	11	2024-08-31	2025-01-01
10	15	2024-10-17	2025-01-01
11	16	2024-11-12	2025-01-01
12	5	2024-12-23	2025-01-01
13	12	2024-12-25	2025-01-01
14	4	2025-01-11	2025-01-01
15	9	2025-01-20	2025-01-01
16	6	2025-03-14	2025-01-01

Explanation :

- **SELECT Event_ID, Start_Date, End_Date**
  Retrieves the event's unique identifier, start date, and end date from the Events table.
- **ORDER BY Start_Date ASC**
  Sorts the results in ascending order based on the Start_Date, ensuring events are listed from earliest to latest.
- **FROM Events**
  Specifies the Events table as the source of the data.

Insights :

- **Broad SQL Support:**
  The ORDER BY clause is available across all major SQL dialects, ensuring consistent behavior in different database systems.
- **Flexible Data Presentation:**
  The ORDER BY clause can be used with one or multiple columns and supports both ascending (ASC) and descending (DESC) order, providing flexibility for various reporting needs.

2.

Connecting Addresses With Outcomes

"As a business analyst, I want to organize data from the Addresses, Event_Outcomes, and Service_Types tables by their respective codes in ascending order so that I can easily analyze relationships and patterns across address IDs, event outcomes, and service types. "

Order By Clause Query :

```
SELECT A.Address_ID,
       A.Address_Details,
       E.Event_Outcome_Code,
       E.Event_Outcome_Description,
       S.Service_Type_Code,
       S.Service_Type_Description
```

```
FROM Addresses A

LEFT JOIN

   Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code

LEFT JOIN

   Ref_service_Types S ON A.Address_ID = S.Service_Type_Code

ORDER BY A.Address_ID ASC,

         E.Event_Outcome_Code ASC,

         S.Service_Type_Code ASC;
```

Results :

Results Messages

	Address_ID	Address_Details	Event_Outcome_Code	Event_Outcome_Description	Service_Type_Code	Service_Type_Description
1	1	616 Little Fleur Place	1	Invalid input provided	1	Workplace Hazard Assessment
2	2	7 Manley Way	2	Invalid input provided	2	Emergency Response Planning
3	3	06750 Delaware Junction	3	Partial success	3	First Aid Training
4	4	9 Jackson Terrace	4	Closed	4	Safety Training
5	5	02429 Carey Street	5	Postponed	5	Safety Training
6	6	5643 Melody Center	6	Invalid input provided	6	Vaccination Drive
7	7	5267 Debra Terrace	7	Connection timeout	7	Fire Safety Inspection
8	8	47738 Spenser Circle	8	Permission denied	8	Vaccination Drive
9	9	61492 Lakeland Point	9	System malfunction	9	Health Checkup
10	10	2319 Londonderry Hill	10	Follow-up Required	10	First Aid Training
11	11	33293 Kingsford Avenue	11	Invalid input provided	11	Emergency Response Planning
12	12	063 Westport Parkway	12	In Progress	12	Emergency Response Planning
13	13	52779 Northwestern Dri...	13	Partial success	13	Emergency Response Planning
14	14	33171 Mayer Crossing	14	Permission denied	14	Health Checkup
15	15	5888 Pearson Point	15	System malfunction	15	Incident Investigation

Explanation :

- **SELECT A.Address_ID, A.Address_Details, E.Event_Outcome_Code, E.Event_Outcome_Description, S.Service_Type_Code, S.Service_Type_Description**
  Retrieves address details, event outcome codes and descriptions, and service type codes and descriptions for analysis.
- **FROM Addresses A**
  Sets the Addresses table as the primary (left) table.
- **LEFT JOIN Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code**
  Joins event outcomes to addresses, keeping all addresses even if there's no matching event outcome.
- **LEFT JOIN Ref_service_Types S ON A.Address_ID = S.Service_Type_Code**
  Joins service types to addresses, again keeping all addresses even if there's no matching service type.
- **ORDER BY A.Address_ID ASC, E.Event_Outcome_Code ASC, S.Service_Type_Code ASC**
  Sorts the results first by address, then by event outcome code, then by service type code, all in ascending order for clear organization.

Insights :

- **Ensures Data Completeness:**
  The query filters for addresses that have both complete event outcome data and documented service types, supporting high data integrity.
- **Supports Compliance and Reporting:**
  By focusing on fully documented records, the query helps ensure compliance with documentation standards, making reporting and analysis more reliable.

2.

Address and Event Outcomes Filter for Safety Service

" As a Safety Program Manager, I want to retrieve a list of addresses with valid event outcome data (excluding invalid inputs) and specific safety-related service types (Safety Training or Fire Safety Inspection) so that I can analyze the effectiveness of these services and allocate resources accordingly. "

Filtering with WHERE Clause, NULL Value Handling, and LEFT JOINs Query :

```
SELECT A.Address_ID,
       A.Address_Details,
       E.Event_Outcome_Description,
       S.Service_Type_Description
FROM   Addresses A
LEFT JOIN
Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code
LEFT JOIN
Ref_Service_Types S ON A.Address_ID = S.Service_Type_Code
WHERE
E.Event_Outcome_Description IS NOT NULL
AND E.Event_Outcome_Description != 'Invalid input provided'
AND (S.Service_Type_Description = 'Safety Training'
OR S.Service_Type_Description = 'Fire Safety Inspection');
```

Results :

Results Messages

	Address_ID	Address_Details	Event_Outcome_Description	Service_Type_Description
1	4	9 Jackson Terrace	Closed	Safety Training
2	5	02429 Carey Street	Postponed	Safety Training
3	7	5267 Debra Terrace	Connection timeout	Fire Safety Inspection
4	16	7628 Gulseth Junction	Successful completion	Fire Safety Inspection

Explanation :

- **FROM Addresses A:**
  Start with the list of addresses.
- **LEFT JOIN Ref_Event_Outcomes E:**
  Add event outcome info for each address (if any).
- **LEFT JOIN Ref_Service_Types S:**
  Add service type info for each address (if any).
- **WHERE:**
  Only keep rows where the event outcome is not empty or invalid.
  Only keep rows where the service type is "Safety Training" or "Fire Safety Inspection".
- **SELECT:**
  Show the address ID, address details, event outcome, and service type.

Insights :

- **Relationship Analysis:**
  By combining data from addresses, event outcomes, and service types, the query enables the identification of patterns, gaps, or relationships across these entities.
- **Enhanced Data Organization:**
  Sorting by address, event outcome code, and service type code makes it easier to review, compare, and analyze the data, supporting more effective business insights and decision-making.

Demo C2: Filtering with WHERE Clause (including handling NULL values)

1.

Complete Address Data Filter for Event Outcomes and Services

“ As a Data Analyst, I want to retrieve a list of addresses that have complete event outcome data and documented service types so that I can ensure data integrity and compliance with documentation standards for reporting and analysis. ”

Filtering with WHERE Clause, NULL Value Handling, and LEFT JOINs Query :

```
SELECT Address_ID,
       Address_Details
FROM   Addresses
WHERE  Address_ID NOT IN (SELECT Event_Outcome_Code FROM
       Ref_Event_Outcomes
WHERE  Event_Outcome_Description IS NULL)
 AND   Address_ID IN (SELECT Service_Type_Code
       FROM Ref_service_Types
       WHERE Service_Type_Description IS NOT NULL);
```

Explanation :

- **SELECT A.Address_ID, A.Address_Details, E.Event_Outcome_Description, S.Service_Type_Description**
  Retrieves address details, event outcome descriptions, and service type descriptions for analysis.
- **FROM Addresses A**
  Uses the Addresses table as the primary data source.
- **LEFT JOIN Ref_Event_Outcomes E ON A.Address_ID = E.Event_Outcome_Code**
  Joins event outcomes to addresses, keeping all addresses even if some have no event outcome.
- **LEFT JOIN Ref_Service_Types S ON A.Address_ID = S.Service_Type_Code**
  Joins service types to addresses, keeping all addresses even if some have no service type.
- **WHERE E.Event_Outcome_Description IS NOT NULL**
  Filters out records where the event outcome description is missing (NULL).
- **AND E.Event_Outcome_Description != 'Invalid input provided'**
  Excludes records with invalid event outcome entries.
- **AND (S.Service_Type_Description = 'Safety Training' OR S.Service_Type_Description = 'Fire Safety Inspection')**
  Includes only addresses associated with the specified safety-related service types.

Insights :

- **Targeted Safety Analysis:**
  The query focuses on addresses linked to key safety services, supporting in-depth evaluation of safety program effectiveness.
- **Data Quality Assurance:**
  By excluding NULL and invalid event outcomes, the query ensures only meaningful, accurate data is analyzed.

Demo C3: Top N Results (Using TOP OR LIMIT /OFFSET)

1.

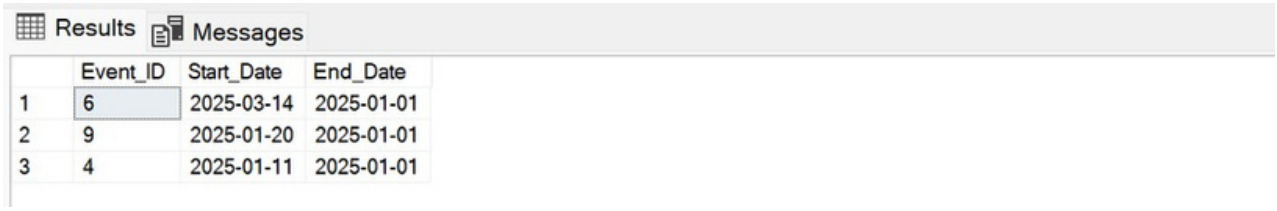
Retrieving Recent Event Records

“ As a Database Analyst, I want to retrieve the top 3 most recent events so that I can review recent activities.”

TOP keyword Query :

```
SELECT TOP 3 Event_ID, Start_Date, End_Date
FROM         Events
ORDER BY     Start_Date DESC;
```

Results :



Results Messages

	Event_ID	Start_Date	End_Date
1	6	2025-03-14	2025-01-01
2	9	2025-01-20	2025-01-01
3	4	2025-01-11	2025-01-01

Explanation :

- **SELECT TOP 3 Event_ID, Start_Date, End_Date**
  Retrieves the Event ID, start date, and end date for only the top 3 records.
- **FROM Events**
  Specifies the Events table as the data source.
- **ORDER BY Start_Date DESC**
  Sorts the events by Start_Date in descending order, so the most recent events appear first.

Insights :

- **Quick Review of Recent Activity:**
  Allows analysts to immediately access and review the latest event records, supporting rapid validation and monitoring.
- **Trend Analysis:**
  Focusing on the most recent events helps in identifying emerging patterns or issues in event scheduling and management.

1a.

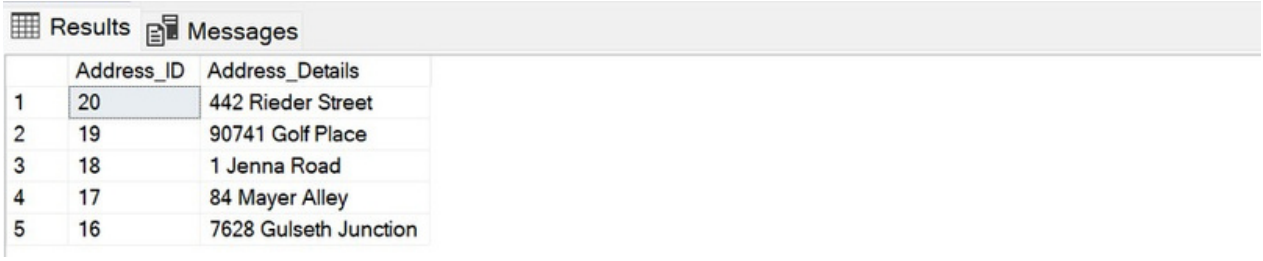
Analyzing Recent Event Data

" As a data analyst, I want to retrieve the 5 most recently added addresses based on their Address_ID so that I can review the latest entries in our system. "

TOP Keyword Query :

```
SELECT TOP 5 Address_ID, Address_Details

FROM        Addresses

ORDER BY    Address_ID DESC;
```

Result :



Results Messages

	Address_ID	Address_Details
1	20	442 Rieder Street
2	19	90741 Golf Place
3	18	1 Jenna Road
4	17	84 Mayer Alley
5	16	7628 Gulseth Junction

Explanation :

- **SELECT TOP 5:**
  This clause limits the result set to the top 5 rows.
  Address_ID, Address_Details:
  Specifies the columns to be retrieved from the Addresses table.
- **FROM Addresses:**
  Specifies the table to retrieve data from.
- **ORDER BY Address_ID DESC:**
  Sorts the addresses in descending order based on their IDs, ensuring that the highest (most recent) IDs are at the top.

Insights :

- **Facilitates Trend Detection:**
  Regularly reviewing the newest records helps identify emerging patterns, such as spikes in new entries or changes in address characteristics.
- **Improves Responsiveness:**
  Quick access to the latest data supports faster decision-making and more agile responses to system changes or business needs.

2.

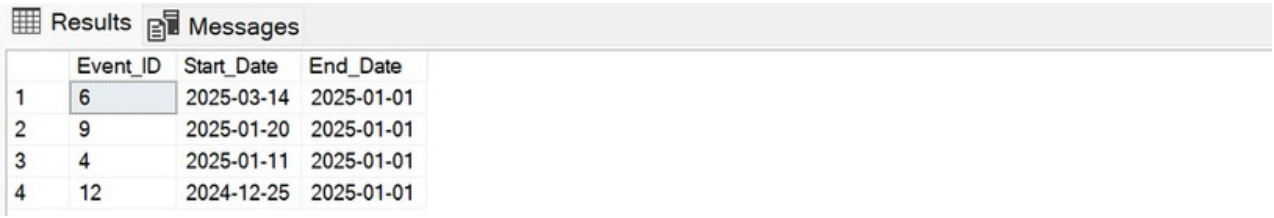
Retrieving the Most Recent Event for Scheduling

“ As an Event Coordinator, I want to retrieve the four most recent events based on their start date so I can quickly see and manage the upcoming schedule. ”

LIMIT /OFFSET Query :

```
SELECT Event_ID,
       Start_Date,
       End_Date
FROM
       Events
ORDER BY
       Start_Date DESC
       OFFSET 0 ROWS FETCH NEXT 4 ROWS ONLY;
```

Results :



Results Messages

	Event_ID	Start_Date	End_Date
1	6	2025-03-14	2025-01-01
2	9	2025-01-20	2025-01-01
3	4	2025-01-11	2025-01-01
4	12	2024-12-25	2025-01-01

Explanation :

- **SELECT Event_ID, Start_Date, End_Date**
  Retrieves the event ID, start date, and end date for each event.
- **FROM Events**
  Specifies the Events table as the data source.
- **ORDER BY Start_Date DESC**
  Sorts the events by start date in descending order, so the most recent events appear first.
- **OFFSET 0 ROWS**
  Skips zero rows, meaning the result starts from the very first row after sorting.
- **FETCH NEXT 4 ROWS ONLY**
  Retrieves only the next four rows, effectively limiting the result set to the four most recent events.

Insights :

- **SQL Server Compatibility:**
  The OFFSET/FETCH syntax is designed for SQL Server (and also supported in some other databases), ensuring the query runs efficiently in that environment.
- **Supports Ongoing Monitoring:**
  Regularly running this query helps coordinators stay updated on the latest events, supporting proactive management and timely decision-making.

2a.

Retrieve a Specific Range of Addresses

"As a database administrator, I need to retrieve a specific set of addresses (from the 6th to the 10th) from the Addresses table, ordered by Address_ID in ascending order, to review the address data in batches for maintenance purposes."

LIMIT /OFFSET Query :

```
SELECT Address_ID, Address_Details

FROM   Addresses

ORDER BY Address_ID ASC

OFFSET 5 ROWS FETCH NEXT 5 ROWS only;
```

Results :

Results Messages

	Address_ID	Address_Details
1	6	5643 Melody Center
2	7	5267 Debra Terrace
3	8	47738 Spenser Circle
4	9	61492 Lakeland Point
5	10	2319 Londonderry Hill

Explanation :

- **SELECT Address_ID, Address_Details**
  Retrieves the address ID and details for each address.
- **FROM Addresses**
  Specifies the Addresses table as the data source.
- **ORDER BY Address_ID ASC**
  Sorts the results in ascending order based on Address_ID, ensuring a consistent and predictable order.
- **OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY**
  Skips the first five rows and then fetches the next five rows, effectively retrieving addresses 6 through 10 in the sorted list. This approach is commonly used for pagination, allowing data to be processed or displayed in manageable batches

Insights :

- **Customizable Data Review:**
  This method allows database administrators to target and review specific ranges of records, supporting maintenance, auditing, and reporting needs.
- **Widely Supported Syntax:**
  While the exact syntax may vary between database systems (e.g., LIMIT/OFFSET in MySQL, FETCH NEXT in SQL Server), the concept of paginating results is broadly supported and essential for scalable data management

Demo C4: Pattern Matching using LIKE.

1.

Identifying Training Events

“ As a Training Coordinator, I want to find all events that have "Training" in their Other_Details field so that I can identify specific training sessions. ”

LIKE Operator Query :

```
SELECT Event_ID,
       Start_Date,
       End_Date,
       Other_Details
FROM   Events
WHERE  Other_Details LIKE '%Training%';
```

Results :

Results Messages

	Event_ID	Start_Date	End_Date	Other_Details
1	1	2024-07-26	2025-01-01	Includes Hands-On training Session
2	4	2025-01-11	2025-01-01	Includes Hands-On training Session
3	5	2024-12-23	2025-01-01	Includes Hands-On training Session
4	10	2024-05-03	2025-01-01	Includes Hands-On training Session
5	13	2024-08-01	2025-01-01	Includes Hands-On training Session

Explanation :

- **SELECT Event_ID, Start_Date, End_Date, Other_Details**
  Retrieves the event ID, start and end dates, and additional details for each event.
- **FROM Events**
  Specifies the Events table as the data source.
- **WHERE Other_Details LIKE '%Training%'**
  Filters for events where the Other_Details contains the word "Training" anywhere in the text.
  - The % symbols are wildcards that match any sequence of characters before or after "Training".

Insights :

- **Targeted Event Identification:**
  The LIKE operator enables flexible pattern matching, allowing coordinators to easily locate all events related to training, regardless of where the word appears in the details.
- **Enhanced Reporting:**
  Facilitates the creation of focused reports on training events, supporting compliance, performance reviews, and strategic decision-making.

2.

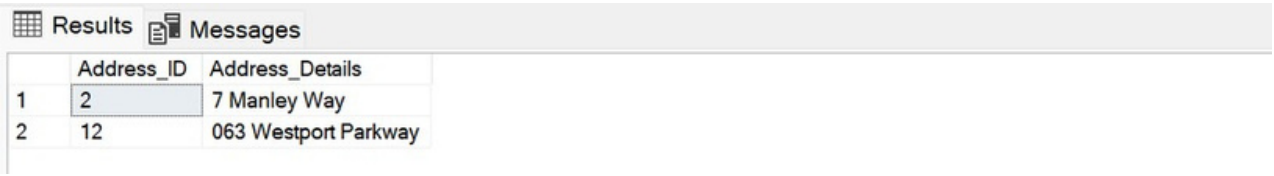
Find Addresses Containing Specific Keywords

“ As a data analyst, I want to find all addresses that contain the word "Way" so I can identify potential residential areas or specific types of streets for targeted marketing or service delivery analysis. ”

LIKE Operator Query :

```
SELECT Address_ID,
       Address_Details
FROM   Addresses
WHERE  Address_Details LIKE '%Way%';
```

Results :


Results Messages

	Address_ID	Address_Details
1	2	7 Manley Way
2	12	063 Westport Parkway

Explanation :

- **SELECT Address_ID, Address_Details**
  Retrieves the unique identifier and detailed address for each record.
- **FROM Addresses**
  Specifies the data source as the Addresses table.
- **WHERE Address_Details LIKE '%Way%'**
  Filters results to include only those addresses where the text "Way" appears anywhere in the Address_Details field.
  % is a wildcard representing any sequence of characters before or after "Way".

Insights :

- **Efficient Location Filtering:**
  The LIKE operator allows analysts to swiftly identify all addresses containing the keyword "Way," supporting focused analysis of street types or residential areas.
- **Flexible Pattern Matching:**
  Using wildcards with LIKE makes the search adaptable, capturing "Way" regardless of its position in the address, which is valuable for comprehensive and accurate reporting.

d. Module 6: Working with Data Types

Why work with appropriate data types?

Using incorrect data types will require you to perform conversions in code, which adds complexity and processing time to your application. Choosing the current data type in your database ensures data integrity, optimizes storage, and allows the database to efficiently process and compare values directly. This reduces the need for client-side data manipulation and improves query performance.

Demo D1: Working with Conversion in a Query

1.

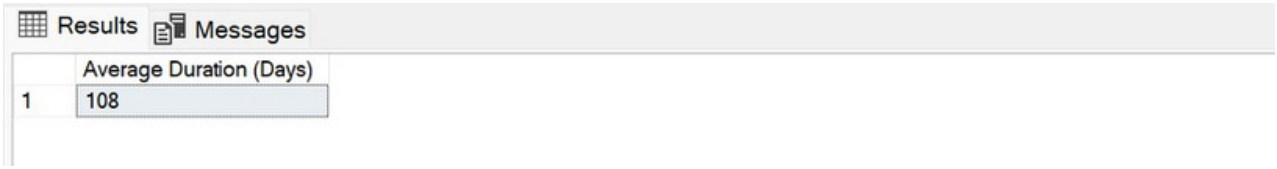
Calculate Average Event Duration

" As a Database Analyst, I want to display the average event duration as an integer so I can easily compare durations without decimal places. "

Cast Function Query :

```
SELECT

AVG(CAST(DATEDIFF(DAY, Start_Date, End_Date) AS INT))

AS "Average Duration(Days)"  FROM Events;
```

Results :



Explanation

- **DATEDIFF(DAY, Start_Date, End_Date):**
  Calculates the number of days between each event's start and end dates.
- **CAST(... AS INT):**
  Converts the calculated duration (which could be a decimal if there are partial days) into an integer, removing any fractional part.
- **AVG(...):**
  Computes the average of all event durations (now as integers).
- **AS "Average Duration(Days)":**
  Assigns a readable label to the resulting column in the output.
- **FROM Events:**
  Specifies that the data comes from the Events table.

Insights :

- **Actionable Metric for Process Improvement:**
  Knowing the average event duration as a whole number allows for straightforward comparisons and benchmarking, helping identify if events are taking longer than expected.
- **Operational Optimization Opportunity:**
  If the average duration is high, it signals potential inefficiencies. This insight can drive initiatives like process redesign, automation, or improved coordination to reduce event timelines and enhance overall efficiency.

2.

Convert Address ID to String

" As a data integrator, I need to convert the Address_ID from an integer to a string so that I can concatenate it with other string-based data in a unified data export for a legacy system. "

Cast Function Query :

```
SELECT
    CAST(Address_ID AS VARCHAR(10)) AS Address_ID_String,
    Address_Details
FROM
    Addresses;
```

Results :

Results Messages

	Address_ID_String	Address_Details
1	1	616 Little Fleur Place
2	2	7 Manley Way
3	3	06750 Delaware Junction
4	4	9 Jackson Terrace
5	5	02429 Carey Street
6	6	5643 Melody Center
7	7	5267 Debra Terrace
8	8	47738 Spenser Circle
9	9	61492 Lakeland Point
10	10	2319 Londonderry Hill
11	11	33293 Kingsford Avenue
12	12	063 Westport Parkway
13	13	52779 Northwestern Drive
14	14	33171 Mayer Crossing

Explanation :

- **CAST(Address_ID AS VARCHAR(10)) AS Address_ID_String:**
  Converts the integer Address_ID to a string (VARCHAR) of up to 10 characters, allowing it to be used in string operations or concatenated with other text fields.
- **Address_Details:**
  Retrieves the address details as-is.
- **FROM Addresses:**
  Specifies the data source table.

Insights :

- **Data Integration and Compatibility:**
  Converting numeric IDs to strings resolves compatibility issues when exporting data to systems or applications that require string-based identifiers, such as legacy platforms.
- **Enhanced Flexibility:**
  This conversion enables easier concatenation and formatting of IDs with other string fields, supporting unified data exports and more flexible reporting

Demo D2: Working with Collation in a Query

1.

Case-Insensitive Search for Training Event Types

“ As a Training Coordinator, I want to search for event types regardless of case sensitivity so that I can find all relevant training programs. ”

COLLATE Function and LIKE Operator Query :

```
SELECT

    Event_Type_Description

FROM

    Ref_Event_Types

WHERE

    Event_Type_Description COLLATE Latin1_General_CI_AI

    LIKE '%training%';
```

Results :

Explanation :

- **SELECT Event_Type_Description**
  Retrieves the event type descriptions from the reference table.
- **FROM Ref_Event_Types**
  Specifies the source table containing event type data.
- **WHERE Event_Type_Description COLLATE Latin1_General_CI_AI LIKE '%training%'**
  - COLLATE Latin1_General_CI_AI applies a case-insensitive and accent-insensitive collation to the column, ensuring the search ignores differences in case and accents.
  - LIKE '%training%' finds any event type description containing the word "training", no matter how it is capitalized or accented.

Insights :

- **Data Consistency and Accuracy:**
  By using a case-insensitive and accent-insensitive search, the query overcomes inconsistencies in data entry, ensuring all relevant training programs are found regardless of how they are stored.
- **Improved User Experience:**
  Users can search for training programs using any capitalization or accent, resulting in a more comprehensive and user-friendly search experience.

2.

Retrieve Health Regulations Mentioning 'Employees' (Case-Insensitive)

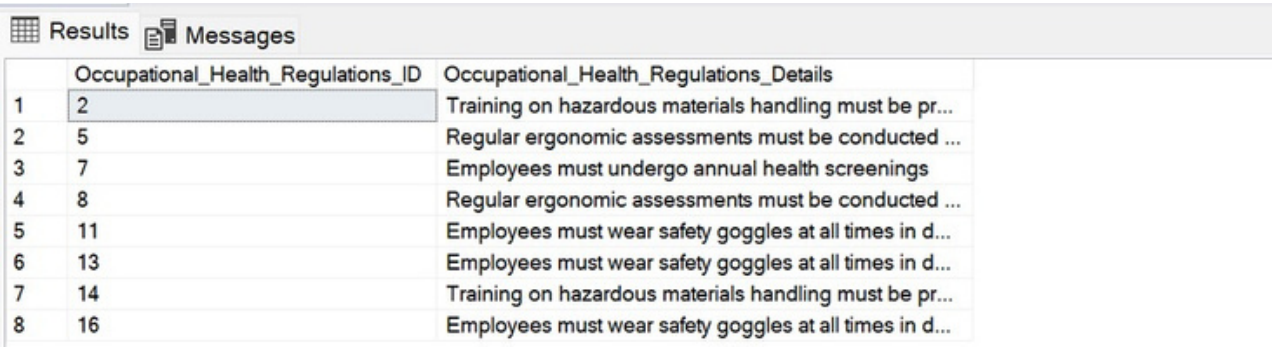
“ As a human resources manager, I need to find all occupational health regulations that mention "employees," regardless of case, so that I can ensure our policies protect our workforce effectively. ”

SELECT with COLLATE and LIKE Query :

```
SELECT

    Occupational_Health_Regulations_ID,

    Occupational_Health_Regulations_Details
```

```
FROM

    Occupational_Health_Regulations

WHERE

    Occupational_Health_Regulations_Details COLLATE

    Latin1_General_CI_AI LIKE '%employees%';
```

Results :



Results | Messages

	Occupational_Health_Regulations_ID	Occupational_Health_Regulations_Details
1	2	Training on hazardous materials handling must be pr...
2	5	Regular ergonomic assessments must be conducted ...
3	7	Employees must undergo annual health screenings
4	8	Regular ergonomic assessments must be conducted ...
5	11	Employees must wear safety goggles at all times in d...
6	13	Employees must wear safety goggles at all times in d...
7	14	Training on hazardous materials handling must be pr...
8	16	Employees must wear safety goggles at all times in d...

Explanation :

- **SELECT Occupational_Health_Regulations_ID, Occupational_Health_Regulations_Details**
  Retrieves the regulation ID and the full text of each occupational health regulation.
- **FROM Occupational_Health_Regulations**
  Specifies the table containing the regulations.
- **WHERE Occupational_Health_Regulations_Details COLLATE Latin1_General_CI_AI LIKE '%employees%'**
  - COLLATE Latin1_General_CI_AI applies a case-insensitive and accent-insensitive collation, ensuring the search ignores case and accent differences.
  - LIKE '%employees%' uses wildcards to match any regulation details containing the word "employees" anywhere in the text, regardless of its position or capitalization.

Insights :

- **Comprehensive Policy Identification:**
  By using a case-insensitive collation, the query ensures all regulations mentioning "employees" are found, regardless of how the word is capitalized, supporting a thorough and accurate compliance review.
- **Data Consistency Across Variations:**
  This approach overcomes inconsistencies in data entry (such as "Employees," "employees," or "EMPLOYEES"), ensuring no relevant regulation is missed due to case differences and improving the reliability of policy analysis

Demo D3: Working with Date and Time Functions

1.

Extracting Event Year

“ As a Health and Safety Officer, I want to extract the year from the event Start_Date so I can analyze trends over time. ”

Scalar function Query :

```
SELECT
    Event_ID,
    Start_Date,
    YEAR(Start_Date) AS "Event Year"
FROM
    Events;
```

Results :

Results Messages

	Event_ID	Start_Date	Event Year
1	1	2024-07-26	2024
2	2	2024-04-18	2024
3	3	2024-08-22	2024
4	4	2025-01-11	2025
5	5	2024-12-23	2024
6	6	2025-03-14	2025
7	7	2024-05-11	2024
8	8	2024-07-11	2024
9	9	2025-01-20	2025
10	10	2024-05-03	2024
11	11	2024-08-31	2024
12	12	2024-12-25	2024
13	13	2024-08-01	2024

Explanation :

- **SELECT Event_ID, Start_Date, YEAR(Start_Date) AS "Event Year"**
  Retrieves the event ID, the original start date, and extracts the year component from the Start_Date column, labeling it as "Event Year".
- **FROM Events**
  Specifies the data source table.

Insights :

- **Year-Based Analysis:**
  Extracting the year from event dates enables trend analysis, annual reporting, and grouping of events by year for summaries or comparisons.
- **Simplified Timelines:**
  Presenting event years alongside IDs and dates provides a clear, high-level view of event distribution over time, making it easier to spot patterns or anomalies.

2.

Find Occupational Health Regulations Updated in the Last Year

" As a compliance officer, I need to find all occupational health regulations that have been updated in the last year so that I can ensure we are following the most current guidelines. "

Date Functions Query :

```
SELECT Occupational_Health_Regulations_ID,

       Occupational_Health_Regulations_Details,

       Regulation_Date

FROM

       Occupational_Health_Regulations

WHERE

       Regulation_Date >= DATEADD(year, -1, GETDATE());
```

Results :

Results Messages

	Occupational_Health_Regulations_ID	Occupational_Health_Regulations_Details	Regulation_Date
1	1	Regular breaks must be taken every 2 hours	2024-06-20
2	2	Training on hazardous materials handling must be pr...	2024-06-19
3	3	First aid kits must be fully stocked and easily accessible	2024-06-18
4	4	Fire drills must be conducted every quarter	2024-06-17
5	5	Regular ergonomic assessments must be conducted ...	2024-06-16
6	6	Proper ventilation must be maintained in work areas	2024-06-15
7	7	Employees must undergo annual health screenings	2024-06-14
8	8	Regular ergonomic assessments must be conducted ...	2024-06-11
9	9	First aid kits must be fully stocked and easily accessible	2024-06-10
10	10	Proper lifting techniques must be followed to prevent i...	2024-06-09
11	11	Employees must wear safety goggles at all times in d...	2024-06-08
12	12	Emergency exits must be clearly marked and accessi...	2024-06-03
13	13	Employees must wear safety goggles at all times in d...	2024-06-04

Explanation :

- **SELECT Occupational_Health_Regulations_ID, Occupational_Health_Regulations_Details, Regulation_Date**
  Retrieves the regulation ID, details, and date from the Occupational_Health_Regulations table.
- **WHERE Regulation_Date >= DATEADD(year, -1, GETDATE())**
  Filters for regulations where the Regulation_Date is within the last year from the current date.
  - GETDATE() returns the current date and time.
  - DATEADD(year, -1, GETDATE()) subtracts one year from the current date to set the lower bound for the date filter.

Insights :

- **Assistance in Monitoring**
  This query helps in monitoring compliance by identifying regulations that have been updated recently, ensuring the organization is following the latest guidelines.
- **Identifying Regulations**
  This provides a way to quickly identify regulations that may require immediate attention or changes to current practices **and** streamlines the process of reviewing regulations by focusing on those that are most recent, saving time and resources.

Demo D4: Working with String Functions

1.

Creating Full Location Description

" As a Database Administrator, I want to concentrate the location name and address to create a full location description so I can standardize location information. "

Concat Function Query :

```
SELECT Location_ID,
    CONCAT(Other_Details, ', ', Address_ID)
    AS "Full Location Description"
FROM Locations;
```

Results :

Results Messages

	Location_ID	Full Location Description
1	1	headquators, 13
2	2	factory floor, 7
3	3	training Area, 7
4	4	medical center, 9
5	5	conference hall, 5
6	6	warehouse, 6
7	7	conference hall, 8
8	8	factory floor, 3
9	9	training Area, 10
10	10	clinic, 2
11	11	headquaters, 16
12	12	training Area, 8
13	13	training Area, 3

Explanation :

- **SELECT Location_ID,**
  Retrieves the unique identifier for each location.
- **CONCAT(Other_Details, ', ', Address_ID) AS "Full Location Description"**
  Uses the CONCAT function to combine the Other_Details (such as location name or description) with the Address_ID, separated by a comma and a space. The result is labeled as "Full Location Description".
- **FROM Locations**
  Specifies the data source table.

Insights :

- **Standardized Location Information:**
  By combining descriptive details with a unique address ID, this approach creates a standardized and comprehensive location label, which is valuable for consistent data entry, reporting, and integration.
- **Enhanced Data Usability:**
  The full location description provides a more informative and user-friendly identifier, making it easier to track, search, and analyze locations across different systems and reports.

2.

Extracting Location Abbreviations

“ As a Data Analyst, I want to extract the first five characters from the Other_Details field of the Locations table to create a standardized abbreviation for each location, so I can use these abbreviations for reporting and quick identification purposes. ”

Substring Function Query :

```
SELECT

    Location_ID,

    SUBSTRING (Other_Details, 1, 5) AS "Abbreviation"

FROM

    Locations;
```

Results :

Results Messages

	Location_ID	Abbreviation
1	1	headq
2	2	facto
3	3	train
4	4	medic
5	5	confe
6	6	wareh
7	7	confe
8	8	facto
9	9	train
10	10	clini
11	11	headq
12	12	train

Explanation :

- **SELECT Location_ID,**
  Retrieves the unique identifier for each location.
- **SUBSTRING(Other_Details, 1, 5) AS "Abbreviation"**
  Uses the SUBSTRING function to extract the first five characters from the Other_Details field, labeling the result as "Abbreviation".
- **FROM Locations**
  Specifies the data source table.

Insights :

- **Concise Location Identifiers:**
  Extracting the first five characters creates short, easily recognizable abbreviations, ideal for reports, dashboards, or any space-limited interfaces.
- **Standardized Naming Convention:**
  This method enforces consistency in how locations are referenced, making data entry, searching, and sorting more efficient and reliable.

4.

Standardizing Location Descriptions

" As a Data Administrator, I want to update the Other_Details field in the Locations table by replacing all instances of "Room" with "Area", ensuring consistent terminology for location types in the database ."

Replace Function Query :

```
UPDATE Locations

SET Other_Details = REPLACE(Other_Details, 'Room', 'Area');
```

Results:

Results | Messages

	Location_ID	Address_ID	Other_Details
1	1	13	headquators
2	2	7	factory floor
3	3	7	training Area
4	4	9	medical center
5	5	5	conference hall
6	6	6	warehouse
7	7	8	conference hall
8	8	3	factory floor
9	9	10	training Area
10	10	2	clinic
11	11	16	headquaters
12	12	8	training Area

Explanation :

- **UPDATE Locations**
  Specifies the table to be updated.
- **SET Other_Details = REPLACE(Other_Details, 'Room', 'Area')**
  Uses the REPLACE function to substitute every occurrence of the word "Room" with "Area" in the Other_Details field for all rows in the table.

Insights :

- **Improved Data Consistency:**
  Standardizing terminology by replacing "Room" with "Area" ensures uniform location descriptions, which enhances data quality and reduces ambiguity in reports and analyses.
- **Simplified Data Management:**
  Consistent naming conventions make it easier to search, filter, and organize location records, streamlining database maintenance and supporting accurate, efficient data usage.

5.

Clean Address Details by Removing Leading/Trailing Spaces

" As a data analyst, I want to remove any leading or trailing spaces from the Address_Details column to ensure data consistency and improve search accuracy. "

Trim Function Query :

```
SELECT Address_ID,

       TRIM(Address_Details) AS Cleaned_Address_Details

FROM   Addresses;
```

Results :

Results Messages

	Address_ID	Cleaned_Address_Details
1	1	123 Main St
2	2	616 Little Fleur Place
3	3	7 Manley Way
4	4	06750 Delaware Junction
5	5	9 Jackson Terrace
6	6	02429 Carey Street
7	7	5643 Melody Center
8	8	5267 Debra Terrace
9	9	47738 Spenser Circle
10	10	61492 Lakeland Point
11	11	2319 Londonderry Hill

Explanation :

- **SELECT Address_ID, TRIM(Address_Details) AS Cleaned_Address_Details**
  Retrieves the address ID and the address details after removing any leading or trailing spaces. The result is labeled as "Cleaned_Address_Details".
- **FROM Addresses**
  Specifies the source table.
- **TRIM(Address_Details)**
  The TRIM function removes all spaces from the start and end of the string in the Address_Details column. This ensures that no extra spaces remain before or after the actual address text.

Insights :

- **Ensures Uniform Data for Accurate Searches:**
  Removing leading and trailing spaces standardizes address entries, which is crucial for accurate searches, comparisons, and reporting.
- **Improves Data Quality and Reliability:**
  Cleaning address data by trimming spaces eliminates errors caused by inconsistent formatting, resulting in more reliable and readable datasets

6.

Append Line Breaks to Occupational Health Regulations Details

" As a report developer, I need to add line breaks to the Occupational_Health_Regulations_Details for improved readability in multi-line reports or text exports."

Char Function Query :

```
SELECT

    Occupational_Health_Regulations_ID,

    Occupational_Health_Regulations_Details

    + CHAR(13) + CHAR(10) AS  Formatted_Details

FROM

    Occupational_Health_Regulations;
```

Results :

Results Messages

	Occupational_Health_Regulations_ID	Formatted_Details
1	1	Regular breaks must be taken every 2 hours
2	2	Training on hazardous materials handling must be pr...
3	3	First aid kits must be fully stocked and easily accessible
4	4	Fire drills must be conducted every quarter
5	5	Regular ergonomic assessments must be conducted ...
6	6	Proper ventilation must be maintained in work areas
7	7	Employees must undergo annual health screenings
8	8	Regular ergonomic assessments must be conducted ...
9	9	First aid kits must be fully stocked and easily accessible
10	10	Proper lifting techniques must be followed to prevent i...
11	11	Employees must wear safety goggles at all times in d...
12	12	Emergency exits must be clearly marked and accessi...
13	13	Employees must wear safety goggles at all times in d...
14	14	Training on hazardous materials handling must be pr...
15	15	Proper lifting techniques must be followed to prevent i...

Explanation :

- **SELECT Occupational_Health_Regulations_ID, Occupational_Health_Regulations_Details + CHAR(13) + CHAR(10) AS Formatted_Details**
  Retrieves the regulation ID and appends a line break (carriage return and line feed) to the end of each regulation detail, labeling the result as "Formatted_Details".
- **FROM Occupational_Health_Regulations**
  Specifies the source table.
- **CHAR(13) + CHAR(10)**
  These are the ASCII codes for carriage return (CR, 13) and line feed (LF, 10). Together, they create a Windows-style new line (CRLF), which is commonly used to insert line breaks in SQL Server and ensures compatibility with text exports and multi-line reports.

Insights :

- **Enhanced Readability for Reports**
  Appending line breaks structures the text for better readability in exported files or multi-line reports, making regulatory details easier to scan and interpret.
- **Improved Formatting for Data Export:**
  Using CRLF (CHAR(13) + CHAR(10)) ensures that line breaks are properly recognized in most Windows-based applications and text editors, supporting clear and organized presentation of information

7.

Safely Format Occupational Health Regulations Table Name

" As a database administrator, I need to dynamically reference the Occupational_Health_Regulations table in SQL queries and want to ensure it's properly escaped to prevent syntax errors or SQL injection. "

Quotename Function Query :

```
SELECT QUOTENAME ('Occupational_Health_Regulations')

AS QuotedTableName;
```

Results :

Explanation :

- **SELECT QUOTENAME('Occupational_Health_Regulations') AS QuotedTableName**
  Wraps the table name in square brackets to ensure it is treated as a valid SQL identifier.

Insights :

- **Prevents Syntax Errors and SQL Injection:**
  Properly escaping table names protects against injection attacks and avoids errors caused by special characters.
- **Ensures Compatibility with Complex Identifiers:**
  Handles table names with spaces or special characters, making dynamic SQL generation safer and more reliable.

e. Module 7: Using DML to Modify Data

Why use Using DML to Modify Data?

DML is an abbreviation for Data Manipulation Language. Represents a collection of programming languages explicitly used to make changes to the data. DML lets you add, update, or delete data in a database. It keeps information accurate and up to date, helping fix errors and add new details as needed. This ensures the data is reliable for decision-making.

Demo E1: Adding Data to Tables

1.

Inserting a New Event into the Events Table

" As a Database Administrator, I want to add a new event to the Events table to reflect newly scheduled activities. "

Insert Statement Query :

```
INSERT INTO
Events (Event_ID, Start_Date, End_Date,
        Location_ID, Event_Outcome_Code,
        Other_Details, Event_Type_Code,
        Occupational_Health_Regulations_ID)
VALUES (101, '2025-04-05', '2025-04-07', 3, 1,
        'Safety Training Event',2, 1);
```

Results :

Results Messages

	Event_ID	Event_Outcome_Code	Event_Type_Code	Location_ID	Occupational_Health_Regulations_ID	Start_Date	End_Date	Other_Details
1	1	3	11	15	1	2024-07-26	2025-01-01	Includes Hands-On training Session
2	2	3	7	4	7	2024-04-18	2025-01-01	On-Session at Main Office
3	3	6	6	10	5	2024-08-22	2025-01-01	Mandatory for all Employees
4	4	16	8	1	3	2025-01-11	2025-01-01	Includes Hands-On training Session
5	5	13	6	6	14	2024-12-23	2025-01-01	Includes Hands-On training Session
6	6	3	2	11	13	2025-03-14	2025-01-01	Follow-Up Required after Completion
7	7	9	14	5	5	2024-05-11	2025-01-01	Mandatory for all Employees
8	8	1	13	12	16	2024-07-11	2025-01-01	Held Virtually Via Teams
9	9	11	8	3	4	2025-01-20	2025-01-01	Mandatory for all Employees
10	10	14	15	15	5	2024-05-03	2025-01-01	Includes Hands-On training Session
11	11	13	13	6	8	2024-08-31	2025-01-01	Follow-Up Required after Completion
12	12	14	5	6	13	2024-12-25	2025-01-01	Follow-Up Required after Completion
13	13	6	14	11	1	2024-08-01	2025-01-01	Includes Hands-On training Session
14	14	2	6	14	11	2024-06-02	2025-01-01	Mandatory for all Employees

Explanation :

- **INSERT INTO Events (columns...) VALUES (values...)**
  Adds a new row to the Events table, specifying values for each listed column (such as Event_ID, Start_Date, End_Date, etc.).

Insights :

- **Ensures Data Completeness:**
  Regularly inserting new events keeps the database current and accurate, supporting reliable scheduling and planning.
- **Enables Effective Event Tracking:**
  Adding new activities allows for comprehensive event tracking and reporting, which is essential for analysis and operational management.

1a.

Add a New Address to the Addresses Table

"As a database administrator, I need to add a new address to the Addresses table when a new location is registered."

Insert Statement Query :

```
INSERT INTO Addresses

        (Address_ID, Address_Details) VALUES (16, '456 Oak

        Avenue');
```

Results :

Before

Results Messages

	Address_ID	Address_Details
1	1	616 Little Fleur Place
2	2	7 Manley Way
3	3	06750 Delaware Junction
4	4	9 Jackson Terrace
5	5	02429 Carey Street
6	6	5643 Melody Center
7	7	5267 Debra Terrace
8	8	47738 Spenser Circle
9	9	61492 Lakeland Point
10	10	2319 Londonderry Hill
11	11	33293 Kingsford Avenue
12	12	063 Westport Parkway
13	13	52779 Northwestern Drive
14	14	33171 Mayer Crossing

After

Results Messages

	Address_ID	Address_Details
11	11	2319 Londonderry Hill
12	12	33293 Kingsford Avenue
13	13	063 Westport Parkway
14	14	52779 Northwestern Dri...
15	15	33171 Mayer Crossing
16	16	5888 Pearson Point
17	17	84 Mayer Alley
18	18	1 Jenna Road
19	19	90741 Golf Place
20	20	442 Rieder Street
21	21	456 Oak Avenue

Explanation :

- **INSERT INTO Addresses (Address_ID, Address_Details) VALUES (16, '456 Oak Avenue')**
  Adds a new row to the Addresses table with the specified ID and address details.

Insights :

- **Supports Database Growth:**
  Inserting new addresses allows the database to expand and stay up-to-date as new locations are registered.
- **Ensures Comprehensive Data Coverage:**
  Regularly adding new address entries guarantees all locations are represented, supporting accurate records and future scalability.

Demo E2: Modifying and Removing Data

1.

Update Event Location

" As a Health and Safety Officer, I want to update the location of an event in the Events table to reflect the correct venue."

Update Statement Query :

```
UPDATE Events SET Location_ID = 5 WHERE Event_ID = 101;
```

Results:

Results Messages

	Event_ID	Event_Outcome_Code	Event_Type_Code	Location_ID	Occupational_Health_Regulations_ID	Start_Date	End_Date	Other_Details
1	1	3	11	15	1	2024-07-26	2025-01-01	Includes Hands-On training Session
2	2	3	7	4	7	2024-04-18	2025-01-01	On-Session at Main Office
3	3	6	6	10	5	2024-08-22	2025-01-01	Mandatory for all Employees
4	4	16	8	1	3	2025-01-11	2025-01-01	Includes Hands-On training Session
5	5	13	6	6	14	2024-12-23	2025-01-01	Includes Hands-On training Session
6	6	3	2	11	13	2025-03-14	2025-01-01	Follow-Up Required after Completion
7	7	9	14	5	5	2024-05-11	2025-01-01	Mandatory for all Employees
8	8	1	13	12	16	2024-07-11	2025-01-01	Held Virtually Via Teams
9	9	11	8	3	4	2025-01-20	2025-01-01	Mandatory for all Employees
10	10	14	15	15	5	2024-05-03	2025-01-01	Includes Hands-On training Session
11	11	13	13	6	8	2024-08-31	2025-01-01	Follow-Up Required after Completion
12	12	14	5	6	13	2024-12-25	2025-01-01	Follow-Up Required after Completion
13	13	6	14	11	1	2024-08-01	2025-01-01	Includes Hands-On training Session
14	14	2	6	14	11	2024-06-02	2025-01-01	Mandatory for all Employees
15	15	14	13	5	10	2024-10-17	2025-01-01	Held Virtually Via Teams
16	16	9	10	14	7	2024-11-12	2025-01-01	On-Session at Main Office
17	101	1	2	5	1	2025-04-05	2025-04-07	Safety Training Event

Explanation :

- **UPDATE Events**
  Specifies the table to modify.
- **SET Location_ID = 5**
  Changes the location of the event to the new location ID (5).
- **WHERE Event_ID = 101**
  Ensures only the event with ID 101 is updated.

Insights :

- **Ensures Accurate Event Tracking:**
  Updating the event location keeps records current, so attendees and resources are directed to the correct venue.
- **Supports Effective Event Management:**
  Maintaining up-to-date location data improves event planning, coordination, and resource allocation.

1a.

Update Address Details for a Specific Address ID

“ As a data entry clerk, I need to update the Address_Details for a specific Address_ID to correct inaccurate information in the database. ”

Update Statement Query :

```
UPDATE Addresses

SET Address_Details = '123 New Main Street' WHERE Address_ID = 1;
```

Results :

Results Messages

	Address_ID	Address_Details
1	1	123 New Main Street
2	2	616 Little Fleur Place
3	3	7 Manley Way
4	4	06750 Delaware Junction
5	5	9 Jackson Terrace
6	6	02429 Carey Street
7	7	5643 Melody Center
8	8	5267 Debra Terrace
9	9	47738 Spenser Circle
10	10	61492 Lakeland Point
11	11	2319 Londonderry Hill
12	12	33293 Kingsford Avenue
13	13	063 Westport Parkway
14	14	52779 Northwestern Dri...
15	15	33171 Mayer Crossing
16	16	5888 Pearson Point

Explanation :

- **UPDATE Addresses**
  Specifies the table to update.
- **SET Address_Details = '123 New Main Street'**
  Sets the new value for the Address_Details column.
- **WHERE Address_ID = 1**
  Limits the update to the row where Address_ID is 1, ensuring only the intended address is modified.

Insights :

- **Ensures Data Accuracy:**
  Correcting address details maintains the reliability and validity of the database, supporting accurate communication and reporting.
- **Enables Targeted Corrections:**
  The use of a specific WHERE clause allows for precise updates, ensuring only the necessary record is changed without affecting unrelated entries.

3.

Delete Canceled Events

"As a Database Analyst, I want to remove canceled events from the Events table to maintain data accuracy. "

Delete Statement Query :

```
DELETE FROM Events

WHERE Other_Details = 'Safety Training Event';
```

Results :

Results Messages

	Event_ID	Event_Outcome_Code	Event_Type_Code	Location_ID	Occupational_Health_Regulations_ID	Start_Date	End_Date	Other_Details
1	1	3	11	15	1	2024-07-26	2025-01-01	Includes Hands-On training Session
2	2	3	7	4	7	2024-04-18	2025-01-01	On-Session at Main Office
3	3	6	6	10	5	2024-08-22	2025-01-01	Mandatory for all Employees
4	4	16	8	1	3	2025-01-11	2025-01-01	Includes Hands-On training Session
5	5	13	6	6	14	2024-12-23	2025-01-01	Includes Hands-On training Session
6	6	3	2	11	13	2025-03-14	2025-01-01	Follow-Up Required after Complet...
7	7	9	14	5	5	2024-05-11	2025-01-01	Mandatory for all Employees
8	8	1	13	12	16	2024-07-11	2025-01-01	Held Virtually Via Teams
9	9	11	8	3	4	2025-01-20	2025-01-01	Mandatory for all Employees
10	10	14	15	15	5	2024-05-03	2025-01-01	Includes Hands-On training Session
11	11	13	13	6	8	2024-08-31	2025-01-01	Follow-Up Required after Complet...
12	12	14	5	6	13	2024-12-25	2025-01-01	Follow-Up Required after Complet...
13	13	6	14	11	1	2024-08-01	2025-01-01	Includes Hands-On training Session
14	14	2	6	14	11	2024-06-02	2025-01-01	Mandatory for all Employees
15	15	14	13	5	10	2024-10-17	2025-01-01	Held Virtually Via Teams
16	16	9	10	14	7	2024-11-12	2025-01-01	On-Session at Main Office

Explanation :

- **DELETE FROM Events**
  Specifies the table (Events) from which records will be deleted.
- **WHERE Other_Details = 'Safety Training Event'**
  Deletes only those rows where the Other_Details column exactly matches 'Safety Training Event'.

Insights :

- **Maintains Data Accuracy:**
  Deleting canceled events ensures that only active and relevant events remain in the database, improving the accuracy of reports and analyses.
- **Prevents Data Clutter:**
  Regularly removing obsolete or canceled records keeps the database clean and efficient, making it easier to manage and query current event data

3.

Remove an Address from the Addresses Table

“ As a database manager, I need to remove an address from the Addresses table when it is no longer valid or required in the system.”

Delete Statement Query :

```
DELETE FROM Addresses

WHERE Address_ID = 21;
```

Results :

Before

Results Messages

	Address_ID	Address_Details
11	11	2319 Londonderry Hill
12	12	33293 Kingsford Avenue
13	13	063 Westport Parkway
14	14	52779 Northwestern Dri...
15	15	33171 Mayer Crossing
16	16	5888 Pearson Point
17	17	84 Mayer Alley
18	18	1 Jenna Road
19	19	90741 Golf Place
20	20	442 Rieder Street
21	21	456 Oak Avenue

After

Results Messages

	Address_ID	Address_Details
10	10	61492 Lakeland Point
11	11	2319 Londonderry Hill
12	12	33293 Kingsford Avenue
13	13	063 Westport Parkway
14	14	52779 Northwestern Dri...
15	15	33171 Mayer Crossing
16	16	5888 Pearson Point
17	17	84 Mayer Alley
18	18	1 Jenna Road
19	19	90741 Golf Place
20	20	442 Rieder Street

Explanation :

- **DELETE FROM Addresses**
  Specifies the Addresses table as the target for deletion.
- **WHERE Address_ID = 21**
  Deletes only the row where the Address_ID is 21, ensuring that only the specified address is removed.

Insights :

- **Supports Data Cleanup:**
  Removing outdated or invalid addresses keeps the database accurate and relevant.
- **Optimizes Storage:**
  Deleting obsolete data reduces unnecessary storage use and maintains an efficient, up-to-date dataset.

Demo E3: Generating Automatic Column Values

1.

Sequence-Based Unique Address_ID Assignment

" As a database administrator, I want to automatically generate unique Address_ID values for new address entries to ensure each record is uniquely identified without manual intervention. "

Sequence Query :

```
CREATE SEQUENCE Address_ID_Seq

START WITH 1 INCREMENT BY 1;
```

Results :

Results Messages

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_c
1	Address_ID_Seq	114099447	NULL	1	0	SO	SEQUENCE_OBJECT	2025-04-08 18:16:27.150	2025-04-

esults Messages

modify_date	is_ms_shipped	is_published	is_schema_published	start_value	increment	minimum_value	maximum_value
2025-04-08 18:16:27.150	0	0	0	1	1	-9223372036854775808	9223372036854775807

is_cycling	is_cached	cache_size	system_type_id	user_type_id	precision	scale	current_value	is_exhausted	last_used_value
0	1	NULL	127	127	19	0	1	0	1

Explanation :

- **CREATE SEQUENCE Address_ID_Seq START WITH 1 INCREMENT BY 1;**
  Creates a sequence object named Address_ID_Seq that starts at 1 and increases by 1 for each new value generated. This sequence can then be used to automatically generate unique Address_ID values for new address entries.

Insights :

- **Automatic Unique ID Generation:**
  Using a sequence ensures each new address receives a unique identifier without manual input, eliminating the risk of duplicate or missing IDs.
- **Reduces Manual Errors and Simplifies Inserts:**
  Automating ID assignment streamlines the insertion process, reduces human error, and supports scalability as the database grows

1a.

Generate Automatic Event_Type_ID Values Using a Sequence

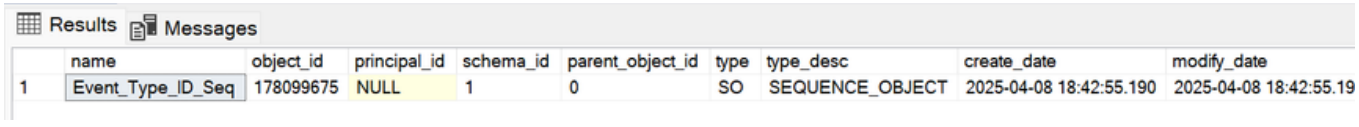
" As a database administrator, I want to auto-generate unique Event_Type_IDs using a sequence to ensure error-free, consistent identifiers. "

Sequence Query :

```
CREATE SEQUENCE Event_Type_ID_Seq

START WITH 1 INCREMENT BY 1;
```

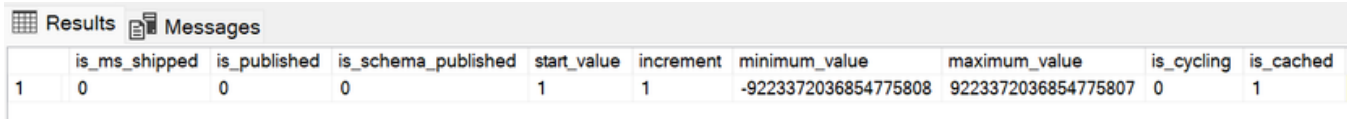
Results :

Creates a sequence for Event_Type_ID



Results Messages

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date
1	Event_Type_ID_Seq	178099675	NULL	1	0	SO	SEQUENCE_OBJECT	2025-04-08 18:42:55.190	2025-04-08 18:42:55.19



Results Messages

	is_ms_shipped	is_published	is_schema_published	start_value	increment	minimum_value	maximum_value	is_cycling	is_cached
1	0	0	0	1	1	-9223372036854775808	9223372036854775807	0	1

Explanation :

- **CREATE SEQUENCE Event_Type_ID_Seq START WITH 1 INCREMENT BY 1;**
  This statement creates a new sequence object named Event_Type_ID_Seq.

- START WITH 1
  Sets the initial value of the sequence to 1.
- INCREMENT BY 1
  Ensures each subsequent value is incremented by 1.
  Sequences are database objects designed to generate unique numeric values, often used for primary keys or identifiers.

Insights :

- **Automatic Unique Identifier Generation:**
  Sequences guarantee that each new event type receives a unique, automatically incremented ID, eliminating manual assignment and reducing the risk of duplicates or errors
- **Supports Multi-User Environments and Scalability:**
  Multiple users can safely generate unique IDs concurrently, making sequences ideal for scalable applications and ensuring data integrity as the system grows

2.

Generate Computed Event_Code Based on Event_Type_Description

" As a database designer, I want to automatically generate a unique Event_Code for each event type based on the Event_Type_ID, ensuring a standardized naming convention across all event types."

Computed Event Query :

```
ALTER TABLE Ref_Event_Types

ADD Event_Code AS ('EVT-' + CAST(Event_Type_Code AS VARCHAR));
```

Results:

Adds Event_Code column to Ref_Event_Types table if it doesn't exist

Results Messages

	Event_Type_Code	Event_Type_Description	Event_Code
1	1	Incident Investigation	EVT-1
2	2	Equipment Maintenance	EVT-2
3	3	Safety Training	EVT-3
4	4	Health Checkup	EVT-4
5	5	First Aid Training	EVT-5
6	6	Workplace Ergonomics Workshop	EVT-6
7	7	Equipment Maintenance	EVT-7
8	8	Health Checkup	EVT-8
9	9	Workplace Ergonomics Workshop	EVT-9
10	10	Mental Health Awareness	EVT-10
11	11	Fire Drill	EVT-11
12	12	Health Checkup	EVT-12
13	13	Hazard Assessment	EVT-13

Explanation :

- **ALTER TABLE Ref_Event_Types**
  Alters the Ref_Event_Types table.
- **ADD Event_Code AS ('EVT-' + CAST(Event_Type_Code AS VARCHAR))**
  Adds a computed column, Event_Code, which concatenates the prefix 'EVT-' with the string version of Event_Type_Code. This creates a standardized event code for each row.

Insights :

- **Automates Standardized Code Generation:**
  Event codes are generated automatically and consistently, removing the need for manual entry and ensuring uniformity.
- **Reduces Maintenance:**
  The computed column updates itself if Event_Type_Code changes, minimizing manual updates and reducing the risk of errors.

2a.

Automatically Generate Event Codes Using Description and Sequence

" As a database designer, I want to automatically generate a unique Event_Code for each event type based on the first three characters of the Event_Type_Description and the Event_Type_ID, ensuring a standardized naming convention across all event types. "

Computed Event Query :

```
ALTER TABLE Ref_Event_Types

ADD Event1_Code AS (

    'EVT-' +

    UPPER(LEFT(Event_Type_Description, 3)) +

    '-' +

    CAST(Event_Type_Code AS VARCHAR(10)));
```

Results :

Adds Event_Code column to Ref_Event_Types table

Results Messages

	Event_Type_Code	Event_Type_Description	Event_Code	Event1_Code
1	1	Incident Investigation	EVT-1	EVT-INC-1
2	2	Equipment Maintenance	EVT-2	EVT-EQU-2
3	3	Safety Training	EVT-3	EVT-SAF-3
4	4	Health Checkup	EVT-4	EVT-HEA-4
5	5	First Aid Training	EVT-5	EVT-FIR-5
6	6	Workplace Ergonomics Workshop	EVT-6	EVT-WOR-6
7	7	Equipment Maintenance	EVT-7	EVT-EQU-7
8	8	Health Checkup	EVT-8	EVT-HEA-8
9	9	Workplace Ergonomics Workshop	EVT-9	EVT-WOR-9
10	10	Mental Health Awareness	EVT-10	EVT-MEN-10
11	11	Fire Drill	EVT-11	EVT-FIR-11
12	12	Health Checkup	EVT-12	EVT-HEA-12
13	13	Hazard Assessment	EVT-13	EVT-HAZ-13
14	14	Mental Health Awareness	EVT-14	EVT-MEN-14
15	15	Hazard Assessment	EVT-15	EVT-HAZ-15
16	16	Fire Drill	EVT-16	EVT-FIR-16

Explanation :

- **ALTER TABLE Ref_Event_Types**
  Alters the Ref_Event_Types table.
- **ADD Event1_Code AS (...):**
  Adds a computed column named Event1_Code using the following formula:
  - 'EVT-' - Prefix to identify the code as an event code.
  - UPPER(LEFT(Event_Type_Description, 3)) - Takes the first three characters of Event_Type_Description and converts them to uppercase.
  - '-' - Hyphen separator.
  - CAST(Event_Type_Code AS VARCHAR(10)) - Converts the numeric Event_Type_Code to a string for concatenation.

Insights :

- **Automatic and Standardized Code Generation:**
  The formula ensures every event code is created automatically and follows a consistent, descriptive naming convention based on event details.
- **Self-Updating and Low Maintenance:**
  The computed column updates itself if either Event_Type_Description or Event_Type_Code changes, reducing manual maintenance and risk of inconsistencies.

f. Module 8: Using Built-In Functions

Why do programmers use built in functions?

TSQL and programming languages use functions. The biggest reasons are functions allow you to do calculation and break programming into more manageable pieces.

Demo F1: Writing Queries with Built-In Functions

1.

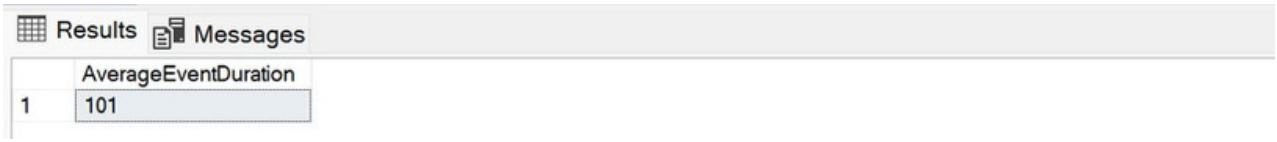
Calculating Average Event Duration

" As a Data Analyst, I want to calculate the average event duration in days to analyze event trends. "

Aggregate Function (AVG) Query :

```
SELECT  AVG (DATEDIFF (DAY, Start_Date, End_Date))

AS AverageEventDuration FROM Events;
```

Results :



Explanation :

- **SELECT AVG(DATEDIFF(DAY, Start_Date, End_Date)) AS AverageEventDuration**
  - DATEDIFF(DAY, Start_Date, End_Date) calculates the number of days between the start and end dates for each event.
  - AVG(...) computes the average duration across all events. The result is labeled as AverageEventDuration.
- **FROM Events**
  Specifies the data source table.

Insights :

- **Simplifies Complex Calculations:**
  Using DATEDIFF and AVG together streamlines the process of calculating average event durations, making analysis efficient and accurate.
- **Supports Data-Driven Planning:**
  Knowing the average event duration helps identify trends and informs better scheduling, resource allocation, and decision-making for future events.

1a.

Extract Street Name from Address Details

" As a data analyst, I want to extract the street name from the Address_Details to normalize the address data and facilitate better reporting and analysis. "

String Extraction Query :

```
SELECT
    Address_ID,
    Address_Details,
    SUBSTRING (Address_Details, 1,
    CHARINDEX (' ', Address_Details) - 1) AS StreetName
FROM
    Addresses
WHERE CHARINDEX(' ', Address_Details) > 0;
```

Results:

Results | Messages

	Address_ID	Address_Details	StreetName
1	1	123 New Main Street	123
2	2	616 Little Fleur Place	616
3	3	7 Manley Way	7
4	4	06750 Delaware Junction	06750
5	5	9 Jackson Terrace	9
6	6	02429 Carey Street	02429
7	7	5643 Melody Center	5643
8	8	5267 Debra Terrace	5267
9	9	47738 Spenser Circle	47738
10	10	61492 Lakeland Point	61492
11	11	2319 Londonderry Hill	2319
12	12	33293 Kingsford Avenue	33293
13	13	063 Westport Parkway	063
14	14	52779 Northwestern Drive	52779
15	15	33171 Mayer Crossing	33171
16	16	5888 Pearson Point	5888
17	17	84 Mayer Alley	84
18	18	1 Jenna Road	1

Explanation :

- **SELECT Address_ID, Address_Details, SUBSTRING(Address_Details, 1, CHARINDEX(' ', Address_Details) - 1) AS StreetName**
  - Retrieves the address ID and full address.
  - Extracts the street name by taking the substring from the start of Address_Details up to (but not including) the first space.
- **FROM Addresses**
  Specifies the table to query.
- **WHERE CHARINDEX(' ', Address_Details) > 0**
  Ensures only addresses containing at least one space are included, preventing errors.

Insights :

- **Enables Structured Data Extraction:**
  Efficiently pulls out street names from address strings, making the data easier to analyze and report on.
- **Improves Data Consistency:**
  Standardizing street name extraction helps normalize address data, supporting more accurate and
  reliable reporting.

Demo F2: Using Conversion Functions

1.

Converting Event IDs to Strings

" As a Database Administrator, I want to convert event IDs into strings for reporting purposes. "

Data Conversion Statement Query :

```
SELECT Event_ID, CONVERT (VARCHAR, Event_ID) AS EventIDString

FROM   Events;
```

Results :

Results Messages

	Event_ID	EventIDString
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

Explanation :

- **SELECT Event_ID, CONVERT(VARCHAR, Event_ID) AS EventIDString FROM Events;**
  - Retrieves the numeric Event_ID and its string equivalent.
  - CONVERT(VARCHAR, Event_ID) is used to change the integer to a string, aliased as EventIDString for reporting purposes.

Insights :

- **Enables Data Type Compatibility:**
  Converting numeric IDs to strings allows seamless integration with systems and reports that require string-based identifiers, ensuring data can be combined and displayed without type conflicts.
- **Improves Reporting Flexibility:**
  Using conversion functions like CONVERT or CAST enhances interoperability and prevents errors when exporting or presenting data in environments that do not support integer-based IDs

2.

Format Regulation Date as MM/DD/YYYY

" As a report generator, I need to display the Regulation_Date in the format MM/DD/YYYY for better readability in reports. "

Converting Date to String Format (Using CONVERT) Query :

```
SELECT
    Occupational_Health_Regulations_ID,
    Occupational_Health_Regulations_Details,
    CONVERT (VARCHAR, Regulation_Date, 101)
    AS Formatted_Regulation_Date
FROM
    Occupational_Health_Regulations;
```

Results :

Results Messages

	Occupational_Health_Regulations_ID	Occupational_Health_Regulations_Details	Formatted_Regulation_Date
1	1	Regular breaks must be taken every 2 hours	06/20/2024
2	2	Training on hazardous materials handling must be pr...	06/19/2024
3	3	First aid kits must be fully stocked and easily accessible	06/18/2024
4	4	Fire drills must be conducted every quarter	06/17/2024
5	5	Regular ergonomic assessments must be conducted ...	06/16/2024
6	6	Proper ventilation must be maintained in work areas	06/15/2024
7	7	Employees must undergo annual health screenings	06/14/2024
8	8	Regular ergonomic assessments must be conducted ...	06/11/2024
9	9	First aid kits must be fully stocked and easily accessible	06/10/2024
10	10	Proper lifting techniques must be followed to prevent i...	06/09/2024
11	11	Employees must wear safety goggles at all times in d...	06/08/2024
12	12	Emergency exits must be clearly marked and accessi...	06/03/2024
13	13	Employees must wear safety goggles at all times in d...	06/04/2024
14	14	Training on hazardous materials handling must be pr...	06/05/2024
15	15	Proper lifting techniques must be followed to prevent i...	06/07/2024
16	16	Employees must wear safety goggles at all times in d...	06/01/2024
17	20	Test Regulation	06/20/2024

Explanation :

- **SELECT Occupational_Health_Regulations_ID, Occupational_Health_Regulations_Details, CONVERT(VARCHAR, Regulation_Date, 101) AS Formatted_Regulation_Date FROM Occupational_Health_Regulations;**
  - Retrieves the regulation ID and details.
  - Uses CONVERT(VARCHAR, Regulation_Date, 101) to format the date as MM/DD/YYYY, labeling it as Formatted_Regulation_Date.

Insights :

- **Improves Readability and Consistency:**
  Formatting dates as MM/DD/YYYY ensures all dates are user-friendly and consistently displayed across reports.
- **Enhances Reporting and Analysis:**
  Presenting dates in a standard format makes it easier for users to interpret, compare, and analyze report data.

3.

Concatenate Regulation Details with Regulation ID as String

"As a report developer, I need to concatenate the Occupational_Health_Regulations_ID with the Occupational_Health_Regulations_Details to create a comprehensive regulation summary."

Converting Data Type (Using CAST) Query :

```
SELECT
    Occupational_Health_Regulations_ID,
    'Regulation ' + CAST (Occupational_Health_Regulations_ID
    AS VARCHAR) + ': ' +
    Occupational_Health_Regulations_Details AS RegulationInfo
FROM
    Occupational_Health_Regulations;
```

Results:

Results Messages

	Occupational_Health_Regulations_ID	RegulationInfo
1	1	Regulation 1: Regular breaks must be taken every 2 ...
2	2	Regulation 2: Training on hazardous materials handl...
3	3	Regulation 3: First aid kits must be fully stocked and ...
4	4	Regulation 4: Fire drills must be conducted every qu...
5	5	Regulation 5: Regular ergonomic assessments must...
6	6	Regulation 6: Proper ventilation must be maintained ...
7	7	Regulation 7: Employees must undergo annual heal...
8	8	Regulation 8: Regular ergonomic assessments must...
9	9	Regulation 9: First aid kits must be fully stocked and ...
10	10	Regulation 10: Proper lifting techniques must be foll...
11	11	Regulation 11: Employees must wear safety goggles...
12	12	Regulation 12: Emergency exits must be clearly mar...
13	13	Regulation 13: Employees must wear safety goggles...
14	14	Regulation 14: Training on hazardous materials han...
15	15	Regulation 15: Proper lifting techniques must be foll...
16	16	Regulation 16: Employees must wear safety goggles...
17	20	Regulation 20: Test Regulation

Explanation :

- **SELECT Occupational_Health_Regulations_ID, 'Regulation ' + CAST(Occupational_Health_Regulations_ID AS VARCHAR) + ': ' + Occupational_Health_Regulations_Details AS RegulationInfo FROM Occupational_Health_Regulations;**
    - Retrieves the regulation ID.
    - Uses CAST(Occupational_Health_Regulations_ID AS VARCHAR) to convert the numeric ID to a string.
    - Concatenates "Regulation ", the ID, a colon, and the details into a single summary string called RegulationInfo.

Insights :

- **Combines Multiple Data Points:**
  Concatenating the ID and details creates a comprehensive summary, making it easier to present and understand regulation information in reports.
- **Improves Report Readability:**
  By merging key details into a single string, the query enhances the clarity and usefulness of report outputs for end users.

Demo F3: Using Logical Functions

1.

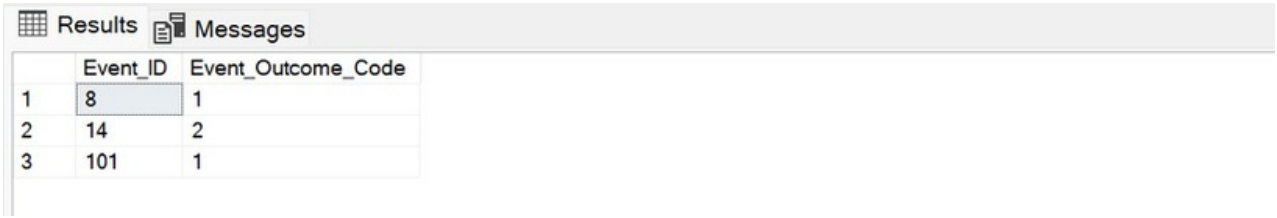
Filtering Events by Outcome Status

" As a Health and Safety Officer, I want to identify events where outcomes are either "Completed" or "Pending". "

Filtering Data with OR operator Query :

```
SELECT Event_ID, Event_Outcome_Code

FROM   Events

WHERE Event_Outcome_Code = 1 OR Event_Outcome_Code = 2;
```

Results :


Results Messages

	Event_ID	Event_Outcome_Code
1	8	1
2	14	2
3	101	1

Explanation :

- **SELECT Event_ID, Event_Outcome_Code FROM Events**
  Specifies the columns (Event_ID, Event_Outcome_Code) to retrieve from the Events table.
- **WHERE Event_Outcome_Code = 1 OR Event_Outcome_Code = 2**
  Uses the **OR** operator to filter rows where Event_Outcome_Code is either 1 (Completed) or 2 (Pending).
  If either condition is true, the row is included in the results.

Insights :

- **Flexible Multi-Condition Filtering:**
  The OR operator allows you to include records that meet any of several conditions, making it easy to retrieve events with different statuses in a single query.
- **Supports Targeted Outcome Tracking:**
  Filtering for both "Completed" and "Pending" events enables focused monitoring and analysis, helping health and safety officers efficiently track event progress and outcomes.

2.

Using the AND and OR Operators

Find Regulations Updated in 2024 Related to Training or Safety

" As a compliance officer, I need to find all regulations updated in 2024 that are related to either "Training" or "Safety" so that I can review them for compliance updates. "

Filtering Data with AND, OR, and LIKE Query :

```
SELECT
    Occupational_Health_Regulations_ID,
    Occupational_Health_Regulations_Details,
    Regulation_Date
FROM
    Occupational_Health_Regulations
WHERE
    YEAR(Regulation_Date) = 2024
    AND (Occupational_Health_Regulations_Details LIKE
    '%Training%' OR Occupational_Health_Regulations_Details
                LIKE '%Safety%');
```

Results :

Results | Messages

	Occupational_Health_Regulations_ID	Occupational_Health_Regulations_Details	Regulation_Date
1	2	Training on hazardous materials handling must be...	2024-06-19
2	11	Employees must wear safety goggles at all times i...	2024-06-08
3	13	Employees must wear safety goggles at all times i...	2024-06-04
4	14	Training on hazardous materials handling must be...	2024-06-05
5	16	Employees must wear safety goggles at all times i...	2024-06-01

Explanation :

- **SELECT Occupational_Health_Regulations_ID, Occupational_Health_Regulations_Details, Regulation_Date** Retrieves the regulation ID, details, and update date for each record.
- **FROM Occupational_Health_Regulations**
  Specifies the source table containing regulation records.
- **WHERE YEAR(Regulation_Date) = 2024**
  Filters records to include only those updated in the year 2024.
- **AND (Occupational_Health_Regulations_Details LIKE '%Training%' OR Occupational_Health_Regulations_Details LIKE '%Safety%')**
  Further filters the results to include only those where the regulation details contain either "Training" or "Safety". The use of parentheses ensures the correct order of evaluation, so both the year and at least one keyword condition must be true.

Insights :

- **Focuses on Specific Criteria**
  Allows you to combine multiple conditions to narrow down results.
- **Enhanced Filtering**
  Provides a more targeted approach to data retrieval.
- **Improved Analysis**
  Simplifies the process of finding relevant regulations for specific compliance needs.

3.

Using the NOT Operator

Find Addresses That Do Not Contain the Word "Street"

"As a data analyst, I want to find all addresses that do not contain the word "Street" to identify and review addresses that might be misclassified."

Filtering Data with NOT LIKE Statement Query :

```
SELECT Address_ID,

       Address_Details

FROM   Addresses

WHERE  NOT (Address_Details LIKE '%Street%');
```

Results :

Results Messages

	Address_ID	Address_Details
1	2	616 Little Fleur Place
2	3	7 Manley Way
3	4	06750 Delaware Junction
4	5	9 Jackson Terrace
5	7	5643 Melody Center
6	8	5267 Debra Terrace
7	9	47738 Spenser Circle
8	10	61492 Lakeland Point
9	11	2319 Londonderry Hill
10	12	33293 Kingsford Avenue
11	13	063 Westport Parkway
12	14	52779 Northwestern Drive
13	15	33171 Mayer Crossing
14	16	5888 Pearson Point
15	17	84 Mayer Alley
16	18	1 Jenna Road
17	19	90741 Golf Place

Explanation :

- **SELECT Address_ID, Address_Details**
  Retrieves the unique address identifier and the address details from the Addresses table.
- **FROM Addresses**
  Specifies the Addresses table as the data source.
- **WHERE NOT (Address_Details LIKE '%Street%')**
  Filters the results to include only those addresses where the Address_Details column does not contain the word "Street".

Insights :

- **Identifies Exceptions**
  Helps you find data that does not conform to a specific pattern.
- **Improves Data Validation**
  Allows you to identify potentially incorrect or misclassified data entries.
- **Enhances Quality Control**
  Provides a way to review addresses that don’t follow a common naming convention.

Demo F4: Using Functions to Work with NULL

1.

Handling NULL Values in Event Details

"As a Data Analyst, I want to replace NULL values in the Other_Details column with "No Details Provided"

Conditional Selection Query :

```
SELECT Event_ID, ISNULL (Other_Details, 'No Details Provided')

AS Other_Details

FROM Events;
```

Results :

Results Messages

	Event_ID	OtherDetails
1	1	Includes Hands-On training Session
2	2	On-Session at Main Office
3	3	Mandatory for all Employees
4	4	Includes Hands-On training Session
5	5	Includes Hands-On training Session
6	6	Follow-Up Required after Completion
7	7	Mandatory for all Employees
8	8	Held Virtually Via Teams
9	9	Mandatory for all Employees
10	10	Includes Hands-On training Session
11	11	Follow-Up Required after Completion
12	12	Follow-Up Required after Completion
13	13	Includes Hands-On training Session
14	14	Mandatory for all Employees
15	15	Held Virtually Via Teams
16	16	On-Session at Main Office
17	101	Safety Training Event

Explanation :

- **SELECT Event_ID**
  Retrieves the unique identifier for each event from the Events table.
- **ISNULL(Other_Details, 'No Details Provided') AS Other_Details**
  Uses the ISNULL function to check if Other_Details is NULL.
  If it is, the function replaces it with the string 'No Details Provided'; otherwise, it returns the original value.
- **FROM Events**
  Specifies the Events table as the source of the data.

Insights :

- **Improved Data Presentation**
  ISNULL helps manage missing data by ensuring that NULL values are replaced with descriptive text, making reports easier to read.
- **Consistent Data Handling**
  By handling NULL values, you guarantee that every event record provides some level of detail, even if specific information is absent.

2.

Using COALESCE() to Select the First Non-NULL Value

"As a database administrator, I want to use COALESCE( ) to select the first non-NULL address detail from multiple columns (in this example, just Address_Details, but the concept extends to multiple columns), providing a default value if all are NULL."

Conditional Selection Query :

```
SELECT Address_ID,

       COALESCE (Address_Details, 'No Address Provided')

       AS Address_Details FROM  Addresses;
```

Results :

Results | Messages

	Address_ID	Address_Details
1	1	123 New Main Street
2	2	616 Little Fleur Place
3	3	7 Manley Way
4	4	06750 Delaware Junction
5	5	9 Jackson Terrace
6	6	02429 Carey Street
7	7	5643 Melody Center
8	8	5267 Debra Terrace
9	9	47738 Spenser Circle
10	10	61492 Lakeland Point
11	11	2319 Londonderry Hill
12	12	33293 Kingsford Avenue
13	13	063 Westport Parkway
14	14	52779 Northwestern Drive
15	15	33171 Mayer Crossing
16	16	5888 Pearson Point
17	17	84 Mayer Alley
18	18	1 Jenna Road
19	19	90741 Golf Place
20	20	442 Rieder Street

Explanation :

- **SELECT Address_ID**
  Retrieves the unique identifier for each address from the Addresses table.
- **COALESCE(Address_Details, 'No Address Provided') AS Address_Details**
  Uses the COALESCE function to return the value of Address_Details if it is not NULL. If Address_Details is NULL, it returns the default string 'No Address Provided' instead.
- **FROM Addresses**
  Specifies the Addresses table as the source of the data.

Insights :

- **Data Resilience**
  Provides a way to handle missing data by selecting from alternative columns or providing default values.
- **Simplified Queries**
  Simplifies queries by consolidating multiple checks into a single function.
- **Flexible Data Handling**
  Allows you to handle data from different sources with varying degrees of completeness.

g. Module 9: Grouping and Aggregating Data:

Why do programmers need to Group and aggregate data?

Programmers need to group and aggregate data because it allows them to summarize large datasets into meaningful information. By grouping data, they can apply aggregate functions like COUNT, SUM, AVG, MIN, and MAX to derive insights. This enables them to identify trends, patterns, and key metrics within the data, which is crucial for making informed decisions and generating reports.

Demo G1: Using GROUP BY Clause and Applying Aggregation:

1.

Analyzing Event Distribution by Location

"As a Data Analyst, I want to calculate the total number of events for each location so that I can analyze event distribution."

Aggregate Query :

```
SELECT Location_ID, COUNT(Event_ID) AS Total_Events

FROM Events GROUP BY Location_ID;
```

Results :

Results Messages

	Location_ID	Total_Events
1	1	1
2	3	2
3	4	1
4	5	2
5	6	3
6	10	1
7	11	2
8	12	1
9	14	2
10	15	2

Explanation :

- **SELECT Location_ID, COUNT(Event_ID) AS Total_Events**
  Retrieves each unique location and counts the number of events associated with it.
- **FROM Events**
  Specifies the Events table as the data source.
- **GROUP BY Location_ID**
  Groups the results by location, so the event count is calculated for each distinct location.

Insights :

- **Category Summarization**
  Using GROUP BY allows you to easily summarize event counts for each location, making comparisons straightforward.
- **Data-Driven Strategies**
  This query supports informed, location-based event strategies by revealing which venues are most or least utilized.

2.

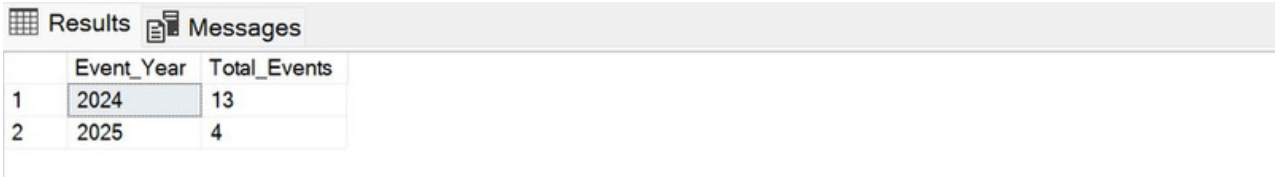
Analyzing Yearly Event Trends

"As a Data Analyst, I want to calculate the total number of events held each year so that I can analyze yearly trends."

Aggregate Statement Query :

```
SELECT

    YEAR (Start_Date) AS Event_Year,
    COUNT (Event_ID) AS Total_Events

FROM

    Events

GROUP BY

    YEAR (Start_Date);
```

Results :


Results Messages

	Event_Year	Total_Events
1	2024	13
2	2025	4

Explanation :

- **SELECT YEAR(Start_Date) AS Event_Year, COUNT(Event_ID) AS Total_Events**
  Extracts the year from each event's start date and counts the total number of events for each year.
- **FROM Events**
  Uses the Events table as the data source.
- **GROUP BY YEAR(Start_Date)**
  Groups the results by the extracted year, so event counts are calculated for each year separately.

Insights :

- **Aggregated Yearly Data**
  Grouping by year allows you to easily see the total number of events held each year for summary analysis.
- **Trend Identification**
  Analyzing event counts by year helps reveal patterns of increase or decrease in activity over time.

2a.

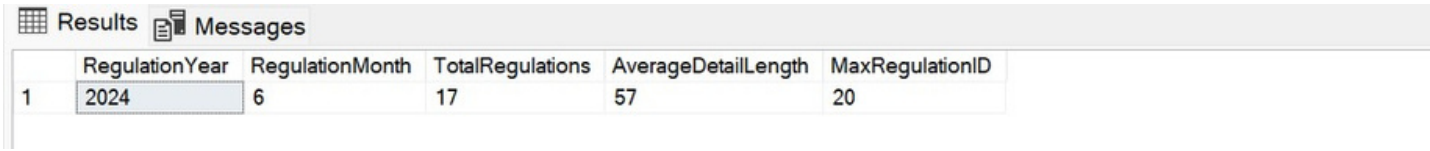
Monthly Regulation Activity Analysis

" As a compliance analyst, I need to understand the monthly trends in occupational health regulations. This includes knowing the total number of regulations, the average length of the regulation details, and the highest regulation ID for each month, so that I can identify patterns, plan audits, and allocate resources effectively. "

Aggregate Statement Query :

```
SELECT
    YEAR(Regulation_Date) AS RegulationYear,
    MONTH(Regulation_Date) AS RegulationMonth,
    COUNT(Occupational_Health_Regulations_ID) AS TotalRegulations,
    AVG(LEN(Occupational_Health_Regulations_Details))
    AS AverageDetailLength,
    MAX(Occupational_Health_Regulations_ID) AS MaxRegulationID
FROM
    Occupational_Health_Regulations
GROUP BY
    YEAR(Regulation_Date), MONTH(Regulation_Date)
ORDER BY
    RegulationYear, RegulationMonth;
```

Results :



Results Messages

	RegulationYear	RegulationMonth	TotalRegulations	AverageDetailLength	MaxRegulationID
1	2024	6	17	57	20

Explanation :

- **SELECT YEAR(Regulation_Date) AS RegulationYear, MONTH(Regulation_Date) AS RegulationMonth**
  Extracts the year and month from each regulation date to enable monthly grouping.
- **COUNT(Occupational_Health_Regulations_ID) AS TotalRegulations**
  Counts the total number of regulations issued each month.
- **AVG(LEN(Occupational_Health_Regulations_Details)) AS AverageDetailLength**
  Calculates the average length of regulation details for each month, indicating detail complexity or thoroughness.
- **MAX(Occupational_Health_Regulations_ID) AS MaxRegulationID**
  Finds the highest regulation ID for each month, which can help track the latest or most recent regulation entered.
- **FROM Occupational_Health_Regulations**
  Specifies the source table containing the regulation data.
- **GROUP BY YEAR(Regulation_Date), MONTH(Regulation_Date)**
  Groups the results by both year and month to enable monthly trend analysis.
- **ORDER BY RegulationYear, RegulationMonth**
  Sorts the output chronologically, making it easier to observe trends over time.

Insights :

- **Trend Identification** Allows you to see how the number of regulations changes from month to month.
- **Workload Management:** Helps in planning workloads by understanding when regulatory activity is highest.
- **Data Quality:** Identifies months with particularly long or short descriptions, which might indicate data quality issues.

Demo G2: the Using HAVING Clause :

1.

Regulation Dates with Long Average Details

"As a compliance analyst, I need to identify the regulation dates on which the average length of the regulation details is greater than 50 characters so that I can focus on dates with more comprehensive or detailed regulatory updates."

Aggregate Query with Filtering (HAVING Clause) :

```
SELECT

    Regulation_Date, AVG (LEN
    (Occupational_Health_Regulations_Details))
    AS Average Detail Length

FROM

    Occupational_Health_Regulations

GROUP BY

    Regulation_Date

HAVING

    AVG(LEN( Occupational_Health_Regulations_Details)) > 50;
```

Result :

Results Messages

	Regulation_Date	AverageDetailLength
1	2024-06-01	67
2	2024-06-03	53
3	2024-06-04	67
4	2024-06-05	74
5	2024-06-07	62
6	2024-06-08	67
7	2024-06-09	62
8	2024-06-10	58
9	2024-06-11	65
10	2024-06-15	51
11	2024-06-16	65
12	2024-06-18	58
13	2024-06-19	74

Explanation :

- **SELECT Regulation_Date, AVG(LEN(Occupational_Health_Regulations_Details)) AS AverageDetailLength:**
  This selects the Regulation_Date and calculates the average length of the regulation details for each date.
- **FROM Occupational_Health_Regulations:**
  Specifies the table from which the data is being retrieved.
- **GROUP BY Regulation_Date:**
  Groups the regulations by the Regulation_Date, so the average length is calculated for each unique date.
- **HAVING AVG(LEN(Occupational_Health_Regulations_Details)) > 50:**
  Filters the grouped results to include only those dates where the average length of the regulation details is greater than 50 characters.

Insights :

- **Comprehensive Change Periods**
  Pinpoints periods when more thorough or complex regulations were issued, indicating times of significant policy activity.
- **Impact Focus**
  Directs attention to dates likely to have the greatest regulatory impact, supporting more efficient compliance management.

Demo G3: Grouping Data by Multiple Columns:

1.

Analyzing Event Outcomes by Location

"As a Database Administrator, I want to count events grouped by both location and event outcome so that I can analyze outcomes per location."

Aggregate Query :

```
SELECT
        Location_ID,  Event_Outcome_Code,
        COUNT (Event_ID) AS Total_Events
FROM
        Events

GROUP BY

       Location_ID, Event_Outcome_Code;
```

Results :

Results Messages

	Location_ID	Event_Outcome_Code	Total_Events
1	3	1	1
2	12	1	1
3	14	2	1
4	4	3	1
5	11	3	1
6	15	3	1
7	10	6	1
8	11	6	1
9	5	9	1
10	14	9	1
11	3	11	1
12	6	13	2
13	5	14	1
14	6	14	1
15	15	14	1
16	1	16	1

Explanation :

- **SELECT Location_ID, Event_Outcome_Code, COUNT(Event_ID) AS Total_Events**
  Retrieves each unique combination of location and event outcome, along with the count of events for each pair.
- **FROM Events**
  Uses the Events table as the data source.
- **GROUP BY Location_ID, Event_Outcome_Code**
  Groups the results by both location and event outcome code, enabling analysis at a more granular level.

Insights :

- **Detailed Category Analysis**
  Grouping by both location and outcome provides a nuanced view of how different outcomes are distributed across locations.
- **High-Impact Location Identification** Helps identify which locations experience the highest number of specific event outcomes, supporting targeted follow-up.

2.

Grouping by Multiple Columns

"As a compliance analyst, I want to group occupational health regulations by Regulation_Date and Event_Type_Description so that I can understand the frequency and distribution of different event types over time."

Aggregate Query with Joins and Grouping :

```
SELECT
    r.Regulation_Date,
    e.Event_Type_Description,
    COUNT(o.Occupational_Health_Regulations_ID) AS RegulationCount
FROM
    Occupational_Health_Regulations o
JOIN
    Ref_Event_Types e ON o.Occupational_Health_Regulations_ID =
e.Event_Type_Code
JOIN
    (SELECT DISTINCT Occupational_Health_Regulations_ID, Regulation_Date
FROM Occupational_Health_Regulations) r
     ON o.Occupational_Health_Regulations_ID =
r.Occupational_Health_Regulations_ID
GROUP BY
    r.Regulation_Date, e.Event_Type_Description
ORDER BY
    r.Regulation_Date, e.Event_Type_Description;
```

Result :

Results Messages

	Regulation_Date	Event_Type_Description	RegulationCount
1	2024-06-01	Fire Drill	1
2	2024-06-03	Health Checkup	1
3	2024-06-04	Hazard Assessment	1
4	2024-06-05	Mental Health Awareness	1
5	2024-06-07	Hazard Assessment	1
6	2024-06-08	Fire Drill	1
7	2024-06-09	Mental Health Awareness	1
8	2024-06-10	Workplace Ergonomics Workshop	1
9	2024-06-11	Health Checkup	1
10	2024-06-14	Equipment Maintenance	1
11	2024-06-15	Workplace Ergonomics Workshop	1
12	2024-06-16	First Aid Training	1
13	2024-06-17	Health Checkup	1
14	2024-06-18	Safety Training	1
15	2024-06-19	Equipment Maintenance	1
16	2024-06-20	Incident Investigation	1

Explanation :

- **COUNT(o.Occupational_Health_Regulations_ID) AS RegulationCount**
  Retrieves the regulation date, event type description, and the count of regulations for each combination.
- **FROM Occupational_Health_Regulations o**
  Uses the Occupational_Health_Regulations table as the main data source.
- **JOIN Ref_Event_Types e ON o.Occupational_Health_Regulations_ID = e.Event_Type_Code**
  Links each regulation to its event type using the event type code.
- **JOIN (SELECT DISTINCT Occupational_Health_Regulations_ID, Regulation_Date FROM Occupational_Health_Regulations) r ON o.Occupational_Health_Regulations_ID = r.Occupational_Health_Regulations_ID**
  Uses a derived table to ensure each regulation-date pair is unique for accurate grouping.
- **GROUP BY r.Regulation_Date, e.Event_Type_Description**
  Groups results by regulation date and event type for aggregation.
- **ORDER BY r.Regulation_Date, e.Event_Type_Description**
  Sorts the output by date and event type for clear, chronological analysis.

Insights :

- **Trend Analysis**
  Helps identify trends in occupational health regulations over time, broken down by event type.
- **Compliance Monitoring**
  Provides insights into which event types are most frequently regulated, aiding in compliance efforts.

SQL Server Practitioner Performance Rating :