Question 01 :

Retrieve a list of users who meet at least one of these criteria:

1. Reputation greater than 8000
2. Created more than 15 posts

Display UserId, DisplayName, and Reputation.

Ensure that each user appears only once in the results.

Question 02 :

Find users who satisfy BOTH of these conditions simultaneously:

1. Have reputation greater than 3000
2. Have earned at least 5 badges

Display UserId, DisplayName, and Reputation.

Question 03 :

Identify posts that have a score greater than 20 but have never received any comments. Display PostId, Title, and Score.

Question 04 :

Create a new permanent table called Posts_Backup that stores all posts with a score greater than 10.
The new table should include: Id, Title, Score, ViewCount, CreationDate, OwnerUserId.

Question 05 :

Create a new table called ActiveUsers containing users who meet the following criteria:

1. Reputation greater than 1000
2. Have created at least one post

The table should include: UserId, DisplayName, Reputation, Location, and PostCount (calculated).

Question 06 :

Create a new empty table called Comments_Template that has the exact same structure as the Comments table but contains no data rows.

Question 07 :

Create a summary table called PostEngagementSummary that combines data from Posts, Users, and Comments tables.
The table should include: PostId, Title, AuthorName, Score, ViewCount CommentCount (calculated), TotalCommentScore (calculated)
Include only posts that have received at least 3 comments.

Question 08 :

Develop a reusable calculation that determines the age of a post in days based on its creation date.
Input: CreationDate (DATETIME)
Output: Age in days (INTEGER)
Test your solution by displaying posts with their calculated ages.

Question 09 :

Develop a reusable calculation that assigns a badge level to users based on their reputation and post activity.
Inputs: Reputation (INT), PostCount (INT)
Output: Badge level (VARCHAR)
Logic:
'Gold' if reputation > 10000 AND posts > 50
'Silver' if reputation > 5000 AND posts > 20
'Bronze' if reputation > 1000 AND posts > 5
'None' otherwise

Question 10 :

Develop a reusable query that retrieves posts created within a specified number of days from today.
Input: @DaysBack (INT) - number of days to look back
Output: Table with PostId, Title, Score, ViewCount, CreationDate
Test with different day ranges (e.g., 30 days, 90 days).

Question 11 :

Develop a reusable query that finds top users from a specific location or all locations based on reputation threshold.
Inputs: @MinReputation (INT), @Location (VARCHAR)
Output: Table with UserId, DisplayName, Reputation, Location, CreationDate
If @Location is NULL, return users from all locations.
Test with different parameters.

Question 12 :

Write a query to find the top 3 highest scoring posts for each PostTypeId.
Use a subquery or CTE with ROW_NUMBER() and PARTITION BY.
Display PostTypeId, Title, Score, and the rank.

Question 13 :

Write a query using a CTE to find all users whose reputation is above the average reputation. The CTE should calculate

1. the average reputation first.
2. Display DisplayName, Reputation, and the average reputation.

Question 14 :

Write a query using a CTE to calculate the total number of posts and average score for each user. Then join with the Users table to display: DisplayName, Reputation, TotalPosts, and AvgScore.
Only include users with more than 5 posts.

Question 15 :

Write a query using multiple CTEs:
First CTE: Calculate post count per user
Second CTE: Calculate badge count per user
Then join both CTEs with Users table to show:
DisplayName, Reputation, PostCount, and BadgeCount.
Handle NULL values by replacing them with 0.

Question 16 :

Write a recursive CTE to generate a sequence of numbers from 1 to 20.
Display the generated numbers.