1. Employee Table Manipulation

Here are the SQL statements for your requests:

1. Create Employee table:

SQL

```
CREATE TABLE Employee (
  SNo INT,
  Name VARCHAR(255),
  Designation VARCHAR(255),
  Branch VARCHAR(255),
  Salary DECIMAL(10,2)  -- Added Salary column
);
```

2. Insert sample data:

SQL

```
INSERT INTO Employee (SNo, Name, Designation, Branch)
VALUES (1, 'Suresh', 'Manager', 'Chennai'),
       (2, 'Ramesh', 'Supervisor', 'Madurai'),
       (3, 'Hari', 'Assistant', 'Trichy');
```

3. Alter table to add Salary column (already done in table creation):

4. Alter table to modify Name column (choose one):

- **Change data type (if needed):**

SQL

```
ALTER TABLE Employee ALTER COLUMN Name VARCHAR(100); -- Reduce size if
needed
```

- **Rename the column:**

SQL

```sql
ALTER TABLE Employee RENAME COLUMN Name TO FullName;
```

5. Describe the table:

SQL

```sql
DESCRIBE Employee;
```

6. Rename table Employee to emp:

SQL

```sql
ALTER TABLE Employee RENAME TO emp;
```

7. Delete the second row:

SQL

```sql
DELETE FROM emp WHERE SNo = 2;
```

8. Drop the table:

SQL

```sql
DROP TABLE emp;
```

2. String Repetition Function

Here's a function to build a string repeatedly:

SQL

```
DELIMITER // -- Define delimiter

CREATE FUNCTION build_string(str VARCHAR(255), count INT)
RETURNS VARCHAR(255)
BEGIN
  DECLARE result VARCHAR(255) DEFAULT '';
  WHILE count > 5 DO
    SET result = CONCAT(result, str);
    SET count = count - 1;
  END WHILE;
  RETURN result;
END // -- End delimiter

DELIMITER ;  -- Reset delimiter

SELECT build_string('Hello ', 10);  -- Example usage
```

This function creates a string by concatenating the input string (str) with itself until count becomes less than or equal to 5.

3. Recursive Factorial Function (MySQL doesn't support recursion natively)`DELIMITER // -- Define delimiter`

```
CREATE PROCEDURE calculate_factorial(IN num INT)

BEGIN

  DECLARE result INT DEFAULT 1;

  DECLARE i INT DEFAULT 1;


  WHILE i <= num DO

    SET result = result * i;
```

```
    SET i = i + 1;
  END WHILE;

  SELECT num, result AS factorial;
END // -- End delimiter

DELIMITER ;  -- Reset delimiter

CALL calculate_factorial(5);  -- Example usage
```

Unfortunately, MySQL doesn't natively support recursion for functions. While there are workarounds involving loops and temporary tables, it's generally not recommended due to performance limitations. Consider using a stored procedure with a loop or a user-defined function written in a language like C or Python and called from MySQL.

4. Stored Procedure to List Employees with Salary > 20000

Here's a stored procedure to list employees with a salary greater than 20000:

SQL

```
DELIMITER // -- Define delimiter

CREATE PROCEDURE get_high_earners()
BEGIN
  SELECT Name, Designation, Salary
  FROM Employee
  WHERE Salary > 20000;
END // -- End delimiter

DELIMITER ;  -- Reset delimiter

CALL get_high_earners();  -- Call the procedure
```

This procedure selects and displays employees from the Employee table where the Salary is

greater than 20000.