Alter table
Designing a database
Foreign key
String functions
NUmerical functions
Date and time function
Other functions

Alter table

```
CREATE TABLE Employees(
    EmployeeID INT,
    FirstName varchar(50),
    LastName varchar(50),
    Department varchar(50),
    Salary Decimal(10,2)
)

ALTER TABLE Employees
ADD Email varchar(100)

ALTER TABLE Employees
ALTER COLUMN LastName varchar(100)

ALTER TABLE Employees
DROP COLUMN Department

ALTER TABLE Employees
ADD PRIMARY KEY (EmployeeID)

ALTER TABLE Employees
ALTER COLUMN EmployeeID INT NOT NULL

ALTER TABLE Employees
ADD CONSTRAINT Salary CHECK (Salary>0)


SELECT * FROM Attendees

INSERT INTO Employees VALUES(1, 'John','Smith', 1000.20, 'john.smith@gmail.com')
INSERT INTO Employees VALUES(2,
'Elizabeth','Taylor',5000.50,'eli.taylor@gmail.com'),(3,
'Alice','Jonas',4000.40,'alice.jonas@gmail.com')

DELETE FROM Employees
WHERE Lastname='Taylor'

EXEC sp_rename Employees, Attendees

DROP TABLE Attendees
```

```
CREATE TABLE BOOKS(
    ID INT,
    Title varchar(100),
    Author varchar(50),
    Year int
)

ALTER TABLE BOOKS
ADD Genre varchar(50)

ALTER TABLE BOOKS
ALTER COLUMN Year SMALLINT

ALTER TABLE BOOKS
ADD PRIMARY KEY (ID)

ALTER TABLE BOOKS
ALTER COLUMN ID INT NOT NULL

ALTER TABLE BOOKS
ADD CONSTRAINT Year CHECK (Year>=1500)

ALTER TABLE BOOKS
DROP COLUMN Author

SELECT * FROM BOOKS;

INSERT INTO BOOKS VALUES(1,'The Great Gatsby',1925, 'Fiction'),(2, 'To Kill a Mockingbird',1960,'Non Fiction'),(3, '1984',1949,'Comedy')

DELETE FROM BOOKS
WHERE ID=2
```

Designing a database

Certainly, I can demonstrate a simplified example of how the Author and Book data might be combined into a single table. However, please note that this approach is not recommended in a real-world scenario, as it can lead to data duplication and other issues. This is just to show you how a single table might look without following proper database design principles.

Example: Single Table Design for Authors and Books (Not Recommended)

ID	Name	BirthDate	Nationality	Title	PublicationYear
1	J.K. Rowling	1965-07-31	British	Harry Potter and...	1997
2	Stephen King	1947-09-21	American	The Shining	1977
3	Stephen King	1947-09-21	American	It	1986

Issues with this Design:

1. **Data Duplication:** Author information is duplicated for each book an author has written. This leads to data redundancy and wastes storage space.
2. **Data Inconsistency:** Multiple instances of the same author's information can lead to inconsistencies if any information needs to be updated.
3. **Lack of Normalization:** The data is not normalized, violating the principles of database design.
4. **Limited Flexibility:** If an author has written many books, the table becomes unwieldy and harder to manage.
5. **Query Performance:** As data grows, querying and maintaining this table becomes inefficient.
6. **Data Integrity:** Ensuring uniqueness of author names or titles becomes challenging.

Remember, while this approach combines data into a single table, it does not follow best practices and will likely cause issues as the database grows. Proper database design involves normalizing data into separate tables and establishing relationships to ensure data integrity and efficient querying.

Foreign key

```
CREATE TABLE CUSTOMERS(
    CustomerID INT PRIMARY KEY,
    CustomerName varchar(50) NOT NULL,
    Email varchar(50) UNIQUE
)

DROP TABLE CUSTOMERS

CREATE TABLE ORDERS(
    OrderID INT PRIMARY KEY,
    CUSTOMERID INT,
    OrderDate DATE,
    TotalAmount DECIMAL(10,2),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
)
```

Joins

```
CREATE TABLE Customers(
    CustomerID INT PRIMARY KEY,
    CustomerName varchar(50) NOT NULL,
    Country varchar(50) NOT NULL
)

CREATE TABLE Orders(
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate Date,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
)

INSERT INTO CUSTOMERS VALUES(1,'John Smith','USA' ),(2, 'Jane Doe','Canada'),(3,
'Micheal Johnson','UK')
```

```
INSERT INTO ORDERS VALUES(101,1,'2023-07-15' ),(102, 2,'2023-07-16'),(103, 3,
'2023-07-17'),(104,3,'2023-07-18'),(105,2,'2023-07-18'
)
```

String functions

```
SELECT CONCAT(ProductName, ' is priced at $', CAST(Price AS VARCHAR(10))) AS
ProductInfo
FROM Products;


SELECT LOWER(ProductName) AS LowercaseProductName
FROM Products;

SELECT LEFT(ProductName, 3) AS ShortProductName
FROM Products;

SELECT ProductName, LEN(ProductName) AS NameLength
FROM Products;

SELECT ProductName, CHARINDEX('p', ProductName) AS PositionOfP
FROM Products;
```

NUmerical functions

```
-- Create a sample table
CREATE TABLE NumberData (
    Value INT
);

-- Insert data into the table
INSERT INTO NumberData (Value) VALUES (4), (7), (12), (5), (9);

-- Calculate SUM, AVG, MAX, and MIN using mathematical functions
SELECT
    SUM(Value) AS Sum,
    AVG(Value) AS Average,
    MAX(Value) AS Maximum,
    MIN(Value) AS Minimum
FROM NumberData;

-- Calculate POWER for each value in the set
SELECT
    Value,
    POWER(Value, 2) AS PowerOfTwo
FROM NumberData;
```

Date and time function

```
-- Create a sample table for events
CREATE TABLE EventData (
    EventName VARCHAR(50),
    EventDate DATETIME
);

-- Insert data into the table
INSERT INTO EventData (EventName, EventDate)
VALUES ('Event A', '2023-08-15 10:00:00'),
       ('Event B', '2023-08-16 14:30:00'),
       ('Event C', '2023-08-17 18:15:00');

-- Find the earliest and latest event dates
SELECT
    MIN(EventDate) AS EarliestEventDate,
    MAX(EventDate) AS LatestEventDate
FROM EventData;

-- Calculate the time difference between events
SELECT
    EventName,
    EventDate,
    LEAD(EventDate) OVER (ORDER BY EventDate) AS NextEventDate,
    DATEDIFF(MINUTE, EventDate, LEAD(EventDate) OVER (ORDER BY EventDate)) AS
TimeToNextEventInMinutes
FROM EventData;
```

Other functions

```
-- Create a sample table
CREATE TABLE UserData (
    UserName VARCHAR(50),
    Age VARCHAR(10)
);

-- Insert data into the table
INSERT INTO UserData (UserName, Age)
VALUES ('John', '30'),
       ('Jane', '25'),
       ('Bob', 'Invalid'),
       ('Alice', NULL);

-- Perform data manipulation using the functions
SELECT
    UserName,
    Age,
    CAST(Age AS INT) AS AgeAsInteger,
    IIF(ISNUMERIC(Age) = 1, 'Numeric', 'Non-Numeric') AS AgeType,
    ISNULL(Age, 'N/A') AS AgeNotNull,
    NULLIF(Age, 'Invalid') AS AgeWithoutInvalid,
    SYSTEM_USER AS CurrentSystemUser,
```

```
    CURRENT_USER AS CurrentDatabaseUser
FROM UserData;
```