2.DDL in MS-SQL Server.

2.1 CREATE DATABASE

The CREATE DATABASE statement creates a new database. The following shows the minimal syntax of the CREATE DATABASE statement:

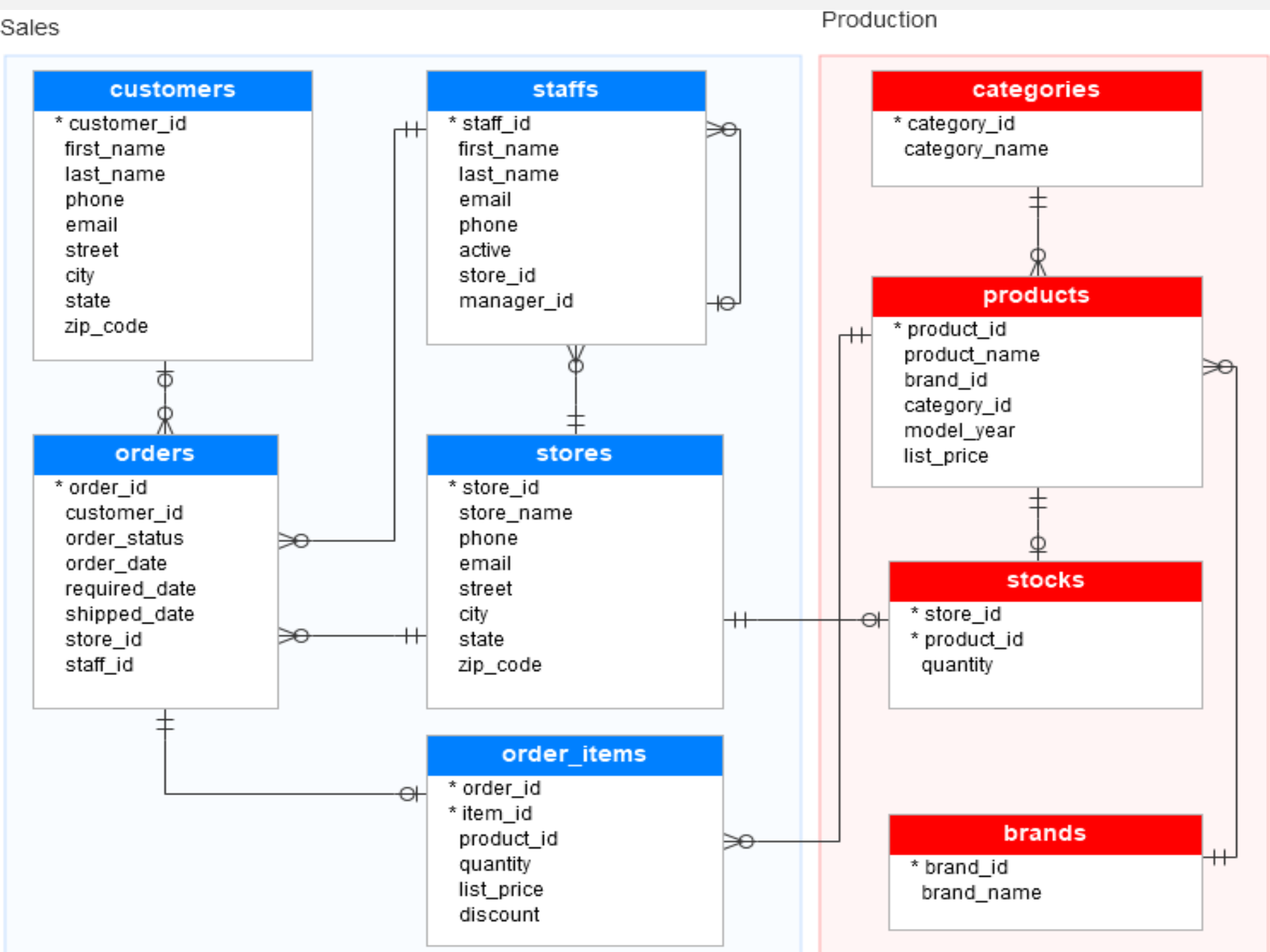
```
CREATE DATABASE database_name;

```

This statement lists all databases in the SQL Server:

```
SELECT
    name
FROM
    master.sys.databases
ORDER BY
    name;
```

```
EXEC sp_databases;
```


Sales
Production
customers
* customer_id
first_name
last_name
phone
email
street
city
state
zip_code
staffs
* staff_id
first_name
last_name
email
phone
active
store_id
manager_id
categories
* category_id
category_name
products
* product_id
product_name
brand_id
category_id
model_year
list_price
orders
* order_id
customer_id
order_status
order_date
required_date
shipped_date
store_id
staff_id
stores
* store_id
store_name
phone
email
street
city
state
zip_code
stocks
* store_id
* product_id
quantity
order_items
* order_id
* item_id
product_id
quantity
list_price
discount
brands
* brand_id
brand_name

2.2 DROP DATABASE

The DROP DATABASE statement allows you to delete one or more databases

with the following syntax:

```
DROP DATABASE  [ IF EXISTS ]
    database_name
    [,database_name2,...];
```

```
DROP DATABASE IF EXISTS TestDb;
```

2.3 CREATE TABLE

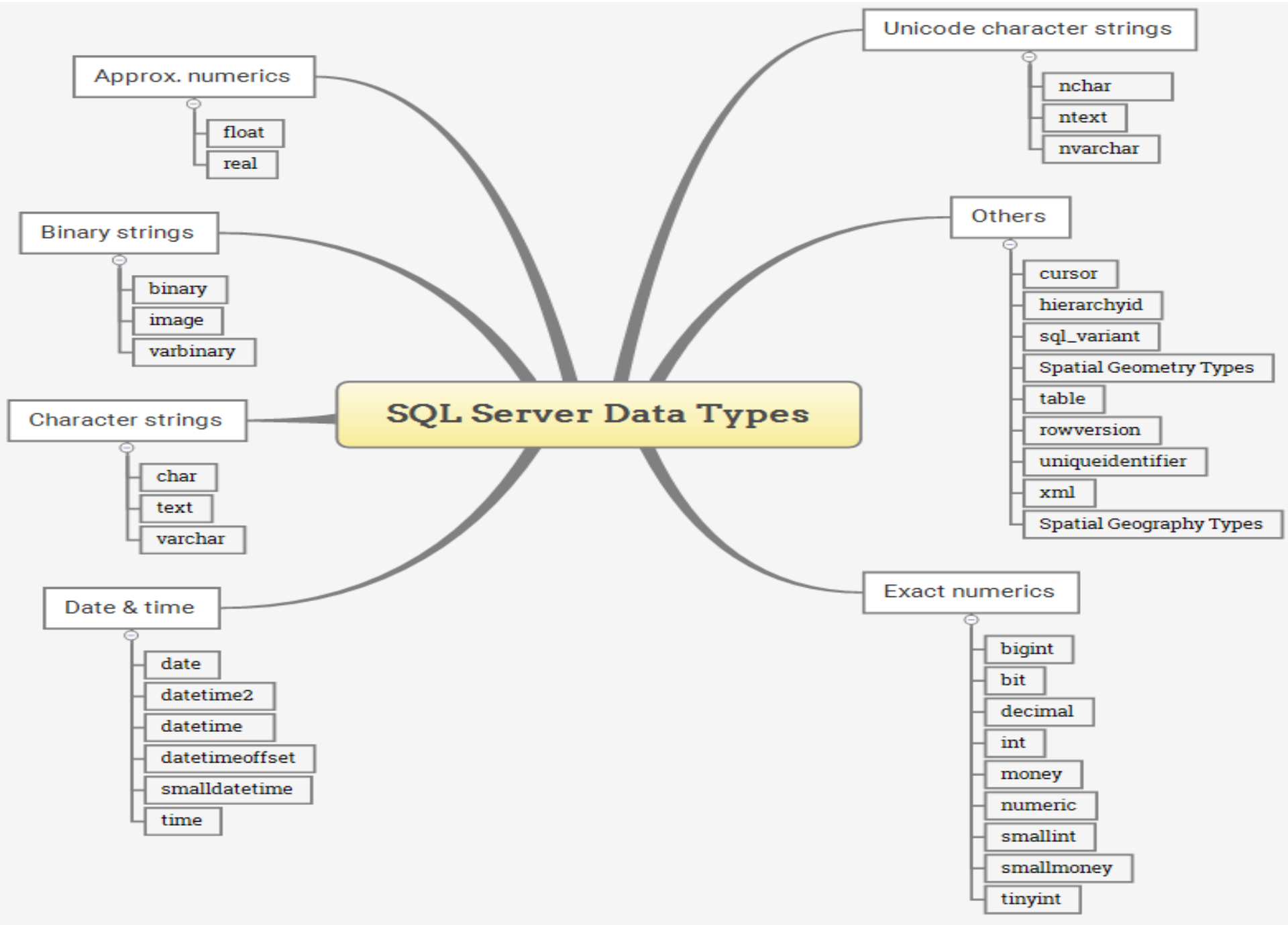
```
CREATE TABLE [database_name.][schema_name.]table_name (
    pk_column data_type PRIMARY KEY,
    column_1 data_type NOT NULL,
    column_2 data_type,
    ...,
    table_constraints
);
```

Identity

```
IDENTITY[(seed,increment)]
```

```
CREATE TABLE sales.stores (
store_id INT IDENTITY (1, 1) PRIMARY KEY,
store_name VARCHAR (255) NOT NULL,
phone VARCHAR (25),
email VARCHAR (255),
street VARCHAR (255),
city VARCHAR (255),
state VARCHAR (10),
zip_code VARCHAR (5)
);
```

2.4 SqlServer Data Types



2.5 Add Column

```
ALTER TABLE table_name
ADD column_name data_type column_constraint;
```

```
CREATE TABLE sales.quotations (
    quotation_no INT IDENTITY PRIMARY KEY,
    valid_from DATE NOT NULL,
    valid_to DATE NOT NULL
);
```

```
ALTER TABLE sales.quotations
ADD description VARCHAR (255) NOT NULL;
```

2.6 Modify Column

Modify column's data type

```
ALTER TABLE table_name
ALTER COLUMN column_name new_data_type(size);
```

2.7 Drop Column

```sql
ALTER TABLE table_name
DROP COLUMN column_name;
```

2.8 Computed Columns

```sql
CREATE TABLE persons
(
    person_id  INT PRIMARY KEY IDENTITY,
    first_name NVARCHAR(100) NOT NULL,
    last_name  NVARCHAR(100) NOT NULL,
    dob        DATE
);
```

```sql
INSERT INTO
    persons(first_name, last_name, dob)
VALUES
    ('John','Doe','1990-05-01'),
    ('Jane','Doe','1995-03-01');
```

```sql
ALTER TABLE persons
ADD full_name AS (first_name + ' ' + last_name);
```

2.9 Rename Table

SQL Rename table using Transact SQL

```
EXEC sp_rename 'old_table_name', 'new_table_name'
```

2.10 Temporary Table – Local | Global

Create temporary tables using **SELECT INTO** statement

```
SELECT
    select_list
INTO
    temporary_table
FROM
    table_name
....
```

```
SELECT
    product_name,
    list_price
INTO #trek_products --- temporary table
FROM
    production.products
WHERE
    brand_id = 9;
```

Global temporary tables: Sometimes, you may want to create a temporary table that is accessible across connections. In this case, you can use global temporary tables.

Unlike a temporary table, the name of a global temporary table starts with a double hash symbol (##).

```
CREATE TABLE ##heller_products (
    product_name VARCHAR(MAX),
    list_price DEC(10,2)
);

INSERT INTO ##heller_products
SELECT
    product_name,
    list_price
FROM
    production.products
WHERE
    brand_id = 3;
```

Constraints

2.11 Primary Key

Introduction to SQL Server **PRIMARY KEY** constraint

```
CREATE TABLE table_name (
    pk_column data_type PRIMARY KEY,
    ...
);
```

2.12 Foreign Key

```
CREATE TABLE sales.staffs (
    staff_id   INT PRIMARY KEY IDENTITY,
    first_name VARCHAR (255) NOT NULL,
    last_name  VARCHAR(50) NOT NULL,
     email      VARCHAR(50) NOT NULL,
     store_id   INT NOT NULL,

 CONSTRAINT fk_staffs_stores FOREIGN KEY (store_id)

        REFERENCES sales.stores(store_id)
);
```

2.13 Check Constraint

The CHECK constraint allows you to specify the values in a column that must satisfy a Boolean expression.

```
CREATE TABLE sales.order_items(
    order_id INT NOT NULL,
    item_id INT NOT NULL,
     order_quantity INT CHECK(order_quantity > 0) NOT NULL,
    unit_price INT CHECK(unit_price > 0) NOT NULL
);
```

2.14 Unique Constraint

SQL Server UNIQUE constraints allow you to ensure that the data stored in a column, or a group of columns, is unique among the rows in a table.

```
CREATE TABLE sales.customers (
    customers_id   INT PRIMARY KEY IDENTITY,
    first_name VARCHAR (255) NOT NULL,
    last_name  VARCHAR(50) NOT NULL,
     email      VARCHAR(50) UNIQUE
);
```

2.15 UNIQUE constraint vs. PRIMARY KEY constraint

❑Although both UNIQUE and PRIMARY KEY constraints enforce the uniqueness of data, you should use the UNIQUE constraint instead of PRIMARY KEY constraint when you want to enforce uniqueness of a column, or a group of columns, that are not the primary key columns.

❑ Different from PRIMARY KEY constraints, UNIQUE constraints allow NULL. Moreover, UNIQUE constraints treat the NULL as a regular value, therefore, it only allows one NULL per column.

2.16 NOT NULL Constraint

Introduction to SQL Server NOT NULL constraint.

The SQL Server NOT NULL constraints simply specify that a column must not assume the NULL.

```
1  CREATE TABLE hr.persons(
2      person_id INT IDENTITY PRIMARY KEY,
3      first_name VARCHAR(255) NOT NULL,
4      last_name VARCHAR(255) NOT NULL,
5      email VARCHAR(255) NOT NULL,
6      phone VARCHAR(20)
7  );
8
9
10
```