ANKARA BİLİM ÜNİVERSİTESİ

Database Management Systems

SQL Data Types Exploration Report MySQL 8.0

Name: Fevzi Samed ÜNAL
ID: 220205032

Department:
Information Systems Engineering

1. Introduction

In SQL-based database systems, data types define what types of values can be stored in each column of a table.

Choosing the correct data type is crucial for data accuracy and performance. Each Database Management System (DBMS) offers its own collection of data types, typically categorized into types such as;

- Numeric Data Types
- String (Character) Data Types
- Binary Data Type
- Date and Time Data Types
- Other Data Type
  - JSON Data Type
  - Spatial Data Type

2. Selected Database Management System

The aim of this assignment is to examine the data types of a selected DBMS, categorize them, and provide SQL examples demonstrating how each data type can be used in a table. **This report focuses on MySQL 8.0,** one of the most widely used SQL database systems. MySQL is one of the most widely used relational database systems in the world and is commonly preferred for web and enterprise applications. It supports a rich set of SQL data types, making it suitable for demonstrating various categories such as Numeric, String, Date and Time Data types.

3. Numeric Data Types (MySQL 8.0)

```sql
CREATE TABLE numeric_example (
  example_tinyint   TINYINT,
  example_smallint  SMALLINT,
  example_mediumint MEDIUMINT,
  example_int       INT,
  example_integer   INTEGER,
  example_bigint    BIGINT,
  example_decimal   DECIMAL(10,2),
  example_numeric   NUMERIC(10,2),
  example_float     FLOAT(5,2),
  example_double    DOUBLE(8,6),
  example_bit       BIT(6)
 );

```

Numeric data types are used to store numerical values such as integers, decimal numbers, and floating-point values. MySQL provides several numeric data types to handle different ranges and precision requirements.

3.1 INT /INTEGER

integers without decimal points. It is commonly used for IDs and counters.

SQL Example INT (CREATE TABLE):

```
CREATE TABLE students (student_id INT, age INT);
```

3.2 TINYINT

Stores have very small integer values. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The size parameter specifies the maximum display width (which is 255)

SQL Example **TINYINT (CREATE TABLE + DEFAULT):**

```
CREATE TABLE user_status (
   is_active TINYINT DEFAULT 1
);
```

3.3 SMALLINT

Stores small integer values with a wider range than TINYINT. A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535.

SQL Example **SMALLINT (INSERT INTO):**

```
INSERT INTO products (stock)
VALUES (150);
```

3.4 BIGINT

A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. Used to store very large integer values such as population counts or transaction IDs.

SQL Example **BIGINT (UPDATE):**

```
UPDATE transactions
SET transaction_id = 9000000001
WHERE transaction_id = 9000000000;
```

3.5 DECIMAL

An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0. Commonly used for financial data.

SQL Example **DECIMAL (DELETE):**

```
DELETE FROM payments
WHERE amount < 10.00;
```

SQL Example **DECIMAL (CREATE INDEX):**

```
CREATE INDEX idx_invoice_total
ON invoices (total);
```

3.6 FLOAT

Stores approximate floating-point numbers. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. Suitable for scientific calculations.

SQL Example **FLOAT (INSERT INTO):**

```
INSERT INTO measurements (temperature)
VALUES (36.5);
```

3.7 DOUBLE

A normal-size floating point number. Stores double-precision floating-point numbers with higher accuracy than FLOAT. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter

SQL Example **DOUBLE (CREATE TABLE):**

```
CREATE TABLE statistics (
   average_score DOUBLE
);
```

3.8 BIT

A bit-value type. The number of bits per value is specified in size. The size parameter can hold a value from 1 to 64. The default value for size is 1. Often used for flags.

SQL Example **BIT (INSERT INTO)**:

```
INSERT INTO settings VALUES (b'1');
```

3.9 MEDIUMINT

SMALLINT is used if it's too small and INT is too much.A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215.

SQL Example **MEDIUMINT (ALTER TABLE):**

```
ALTER TABLE sensors
ADD COLUMN sensor_value MEDIUMINT;
```

3.10 BOOLEAN

Zero is considered false, nonzero values are considered as true.

SQL Example **BOOLEAN (CREATE TABLE):**

```
CREATE TABLE accounts (
   is_verified BOOLEAN
);
```

4. String Data Types (MySQL 8.0)

```sql
CREATE TABLE string_example (
  c_char       CHAR(10),
  c_varchar    VARCHAR(100),
  c_tinytext   TINYTEXT,
  c_text       TEXT,
  c_mediumtext MEDIUMTEXT,
  c_longtext   LONGTEXT,
  c_enum       ENUM('pending','shipped','delivered'),
  c_set        SET('email','sms','push')
);

```

String data types are used to store textual data such as names, descriptions, and identifiers. MySQL 8.0 provides several character data types to handle different text length and storage requirements.

4.1 CHAR

A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1. Best for values with constant length.

SQL Example **CHAR (CREATE TABLE):**

```
CREATE TABLE countries (
  country_code CHAR(2)
);
```

4.2 VARCHAR

Most commonly used string type.A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535

SQL Example **VARCHAR (INSERT INTO):**

```
INSERT INTO users (username)
VALUES ('student01');
```

4.3 TEXT

Stores long text data such as descriptions or comments. Holds a string with a maximum length of 65,535 bytes

SQL Example **TEXT (SELECT):**

```
SELECT description
FROM products;
```

4.4 TINYTEXT

Stores have very small text values. Holds a string with a maximum length of 255 characters

SQL Example **TINYTEXT (ALTER TABLE):**

```
ALTER TABLE notes
ADD COLUMN short_note TINYTEXT;
```

4.5 MEDIUMTEXT

Used for medium-length text such as articles or blog content. Holds a string with a maximum length of 16,777,215 characters

SQL Example **MEDIUMTEXT (UPDATE):**

```
UPDATE articles
SET content = 'Updated article content'
WHERE article_id = 1;
```

4.6 LONGTEXT

Stored very large text data such as documents. Holds a string with a maximum length of 4,294,967,295 characters .

SQL Example **LONGTEXT (DELETE):**

```
DELETE FROM documents
WHERE content IS NULL;
```

4.7 ENUM (val1, val2, val3, ...)

Stores one value from a predefined list of strings. A string object that can have only one value, you can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them

SQL Example **ENUM — (CREATE TABLE):**

```
CREATE TABLE orders (
   status ENUM('pending', 'shipped', 'delivered')
);
```

4.8 SET (val1, val2, val3, ...)

Stores multiple values selected from a predefined list. A string object that can have 0 or more values. You can list up to 64 values in a SET list

SQL Example **SET (UPDATE):**

```
UPDATE user_preferences
SET preferences = 'email,sms'
WHERE user_id = 5;
```

5. Binary Data Types (MySQL 8.0)

```sql
CREATE TABLE binary_example (
  c_binary     BINARY(16),
  c_varbinary  VARBINARY(64),
  c_tinyblob   TINYBLOB,
  c_blob       BLOB,
  c_mediumblob MEDIUMBLOB,
  c_longblob   LONGBLOB
);

```

Binary data types are used to store raw binary data such as images, files, audio, or encrypted content. Unlike string data types, binary data is stored as bytes and is not associated with character sets.

5.1 BINARY

Stores fixed-length binary data as raw bytes. Equal to CHAR () but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1

SQL Example **BINARY (CREATE TABLE):**

```
CREATE TABLE file_hashes (
   hash BINARY(16)
);
```

5.2 VARBINARY

Stores variable-length binary data. Equal to VARCHAR () but stores binary byte strings. The size parameter specifies the maximum column length in bytes.

SQL Example **VARBINARY (INSERT INTO):**

```
CREATE TABLE file_hashes (
   hash VARBINARY(16)
);
INSERT INTO file_hashes (hash)
VALUES (UNHEX('A1B2C3D4'));
```

5.3 BLOB

Stores binary large objects such as files or images.For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data

SQL Example **BLOB (INSERT INTO):**

```
INSERT INTO files (file_data)
VALUES (LOAD_FILE('/path/to/file.bin'));
```

5.4 TINYBLOB

For BLOBs (Binary Large Objects). Max length: 255 bytes

SQL Example **TINYBLOB (CREATE TABLE):**

```
CREATE TABLE thumbnails (
   image_data TINYBLOB
);
```

5.5 MEDIUMBLOB

Stores medium-sized binary data (up to 16 MB).For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data

SQL Example **MEDIUMBLOB (ALTER TABLE):**

```
ALTER TABLE media

ADD COLUMN video_data MEDIUMBLOB;
```

5.6 LONGBLOB

For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data

SQL Example **LONGBLOB (SELECT):**

```
SELECT video_data

FROM media;
```

6.Date & Time Data Types (MySQL 8.0)

```
CREATE TABLE datetime_example (
  col_date      DATE,
  col_time      TIME,
  col_datetime  DATETIME,
  col_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  col_year      YEAR
    );

INSERT INTO datatime_example  VALUES ( '2025-12-13',
                                       '2025-12-13 13:12:39',
                                       CURRENT_TIMESTAMP
                                       '13-12-39',
                                       2025);
```

6.1 DATE

Stores date values in YYYY-MM-DD format. The supported range is from '1000-01-01' to '9999-12-31'

SQL Example **DATE (CREATE TABLE):**

```
CREATE TABLE events (
  event_date DATE
);
```

6.2 TIME

Stores time values in HH:MM:SS format.A time. The supported range is from '-838:59:59' to '838:59:59'

SQL Example **TIME (INSERT INTO):**

```
INSERT INTO schedules (start_time)
VALUES ('09:30:00');
```

6.3 DATETIME

A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time.

SQL Example **DATETIME (UPDATE)**

```
UPDATE meetings
SET meeting_time = '2025-01-15 14:00:00'
WHERE meeting_id = 2;
```

6.4 TIMESTAMP

Stores date and time values and are commonly used for record creation times.

SQL Example **TIMESTAMP (SELECT):**

```
SELECT created_at
FROM users;
```

6.5 YEAR

A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000.

MySQL 8.0 does not support year in two-digit format.

SQL Example **YEAR (ALTER TABLE):**

```
ALTER TABLE employees
ADD COLUMN birth_year YEAR;
```

7. Other Data Types (MySQL 8.0)

7.1 JSON

```sql
CREATE TABLE json_example (
    id  INT PRIMARY KEY AUTO_INCREMENT,
    user_data JSON
    settings JSON,
    profile JSON
);

INSERT INTO json_example (profile)
VALUES ('{"name":"Fevzi","age":22,"city":"Ankara"}'
        '{"name":"Samed","age":22,"city":"Ankara"}'
        '{"name":"Ünal","age":22,"city":"Ankara"}')
;
```

Stores data in JSON (JavaScript Object Notation) format.

Allows structured and semi-structured data storage.

7.2 Spatial Data Types

```
CREATE TABLE spatial_demo (
  g_geometry           GEOMETRY,
  g_point              POINT,
  g_linestring         LINESTRING,
  g_polygon            POLYGON,
  g_multipoint         MULTIPOINT,
  g_multilinestring    MULTILINESTRING,
  g_multipolygon       MULTIPOLYGON,
  g_geometrycollection GEOMETRYCOLLECTION
);

INSERT INTO spatial_demo (
  g_geometry,
  g_point,
  g_linestring,
  g_polygon,
  g_multipoint,
  g_multilinestring,
  g_multipolygon,
  g_geometrycollection
)
VALUES (
  ST_GeomFromText('GEOMETRYCOLLECTION(POINT(1 1))'),
  ST_GeomFromText('POINT(10 20)'),
  ST_GeomFromText('LINESTRING(0 0, 10 10, 20 25)'),
  ST_GeomFromText('POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))'),
  ST_GeomFromText('MULTIPOINT((1 1), (2 2), (3 3))'),
  ST_GeomFromText('MULTILINESTRING((0 0, 5 5), (10 10, 20 20))'),
  ST_GeomFromText('MULTIPOLYGON(((0 0, 0 5, 5 5, 5 0, 0 0)))'),
  ST_GeomFromText(
    'GEOMETRYCOLLECTION(
        POINT(5 5),
        LINESTRING(1 1, 2 2)
     )'
  )
);
```

Used to store geographic or spatial coordinates.

Conclusion

In this report, SQL data types supported by MySQL 8.0 were examined in a structured and comprehensive manner. The study covered

1. Numeric,
2. String(character),
3. Binary,
4. Date and time,
5. Other (JSON and Spatial data types).

Each data type was presented with at least one relevant SQL statement to demonstrate its correct usage within real database operations.

The use of different SQL statements such as CREATE TABLE, INSERT, SELECT, UPDATE, DELETE, ALTER TABLE, and indexing operations demonstrated practical knowledge beyond basic table creation.

As a result, this assignment provides a clear overview of MySQL 8.0 data types and illustrates how they are correctly applied in SQL queries. Understanding these data types and their proper usage is essential for effective database design and reliable application development.