RDBMS and MySQL

What is Database?

A **database** is a structured representation of data that we can read from and write to where structured means data is stored in tables, columns and rows.

What is DBMS?

A **DBMS** (database management system) is a software system that stores, manipulates and manages a database.

Types of DBMS:

1. Relational DBMS (RDBMS): Uses tables
2. NoSQL: Stores in key-value pairs, documents and graphs

Table:

1. Table (Relation): Grouping of data in a relational database.
2. Rows (Record/Tuple): A single, logical item in a table.
3. Column (Field/Attribute): A fact to be tracked in a table.

Constraints:

1. A **Primary Key** uniquely identifies each row in a table.
2. The **Unique** constraint ensures that all values in a column are distinct. Unlike the primary key, it allows NULL values.
3. The **Not null** constraint ensures that a column cannot store NULL values.

ACID properties:

1. Atomicity: The entire transaction takes place at once or not at all
2. Consistency: The database must be consistent before and after the transaction
3. Isolation: Multiple transactions occur independently without interference
4. Durability: The changes of a successful transaction occurs even if system fails

Entity Relationship Diagrams (ERD): A visual representation of the database structure.

What is SQL?

Structured Query Language designed to work with RDBMS

What is Query?

A statement that retrieves data from a database

Syntax:

```
select <col_name>
from <tbl_name>
where <conditions>
group by <col_name>
having <conditions>
order by <col_name>;
```

DDL: Data Definition Language

1. **Create:**

```
CREATE TABLE employees (id INT, name VARCHAR(50), salary INT);
```

2. **Alter:**

```
ALTER TABLE employees ADD COLUMN department VARCHAR(50);
```

3. **Drop:**

```
DROP TABLE employees;
```

DML: Data Manipulation Language

1. **Insert:**

```
INSERT INTO employees (id, name, salary) VALUES (1, 'John', 50000);
```

2. **Update:**

```
UPDATE employees SET salary = 55000 WHERE id = 1;
```

3. **Delete:**

```
DELETE FROM employees WHERE id = 1;
```

Cloning a Table (Deep Copy vs. Shallow Copy)

1. **Deep Copy**: Copies both table structure and the data

```sql
CREATE TABLE new_table AS SELECT * FROM old_table;
```

2. **Shallow Copy:** Copies only the table structure but not the data

```sql
CREATE TABLE new_table LIKE old_table;
```

DESC Command:

```sql
DESC employees;
```

Views:

```sql
CREATE VIEW high_salary AS
SELECT * FROM employees WHERE salary > 60000;
```

SELECT Statement:

Syntax:

```sql
SELECT ColumnName1, ColumnName2, ColumnNameX
FROM TableName;
```

Select columns from table

```sql
SELECT DateReceived, Product, Company, State
FROM Complaint;
```

Select all columns from table:

```sql
SELECT *
FROM Complaint;
```

WHERE Condition Statement:

```sql
SELECT DateReceived, Product, Issue, Company
FROM Complaint
WHERE State = 'LA';
```

Expression	Usage	Example
=	Equals	`ComplaintId = 1653822`
!= , <>	Not equals	`ComplaintId != 1653822` `ComplaintId <> 1653822`
>, >=	Greater than, Greater than or equal to	`ComplaintId > 10000` `ComplaintId >= 10000`
<, <=	Less than, Less than or equal to	`ComplaintId < 10000` `ComplaintId <= 10000`
BETWEEN	Column value in an **inclusive** range.	`ComplaintId BETWEEN 1000 AND 30000`

LIKE:

Expression	Description
LIKE 'A%'	Matches strings that start with the letter 'A'. (case insensitive by default)
LIKE 'a%c'	Matches strings that start with 'a', end with 'c', and have any number of characters in between.
LIKE '%space%'	Matches strings that contain the value 'space' anywhere.
LIKE '%'	Matches all strings. Therefore, it's not particularly useful.
LIKE '_at'	Matches strings that start with any single character and end with 'at'.
LIKE '___'	Matches any string exactly three characters long.