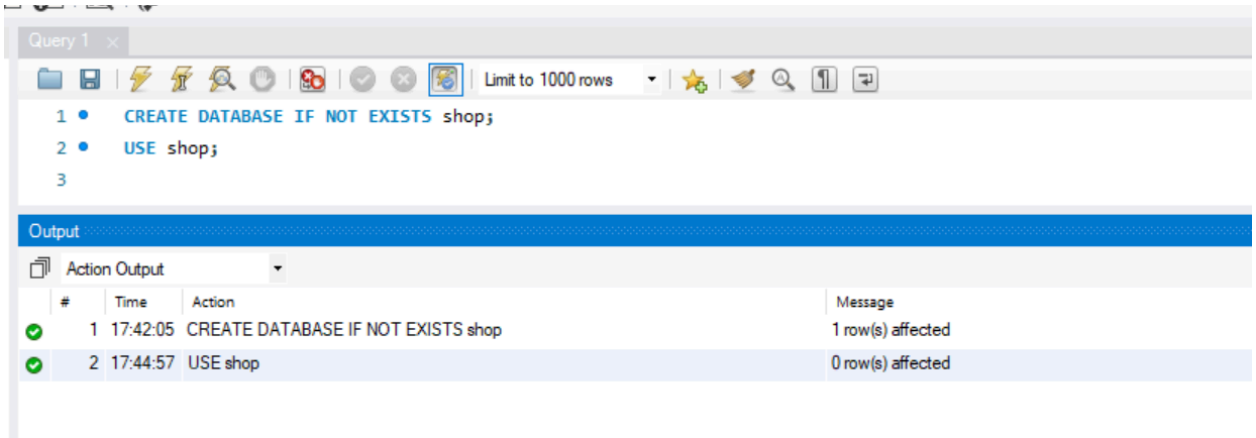
1. Need for database

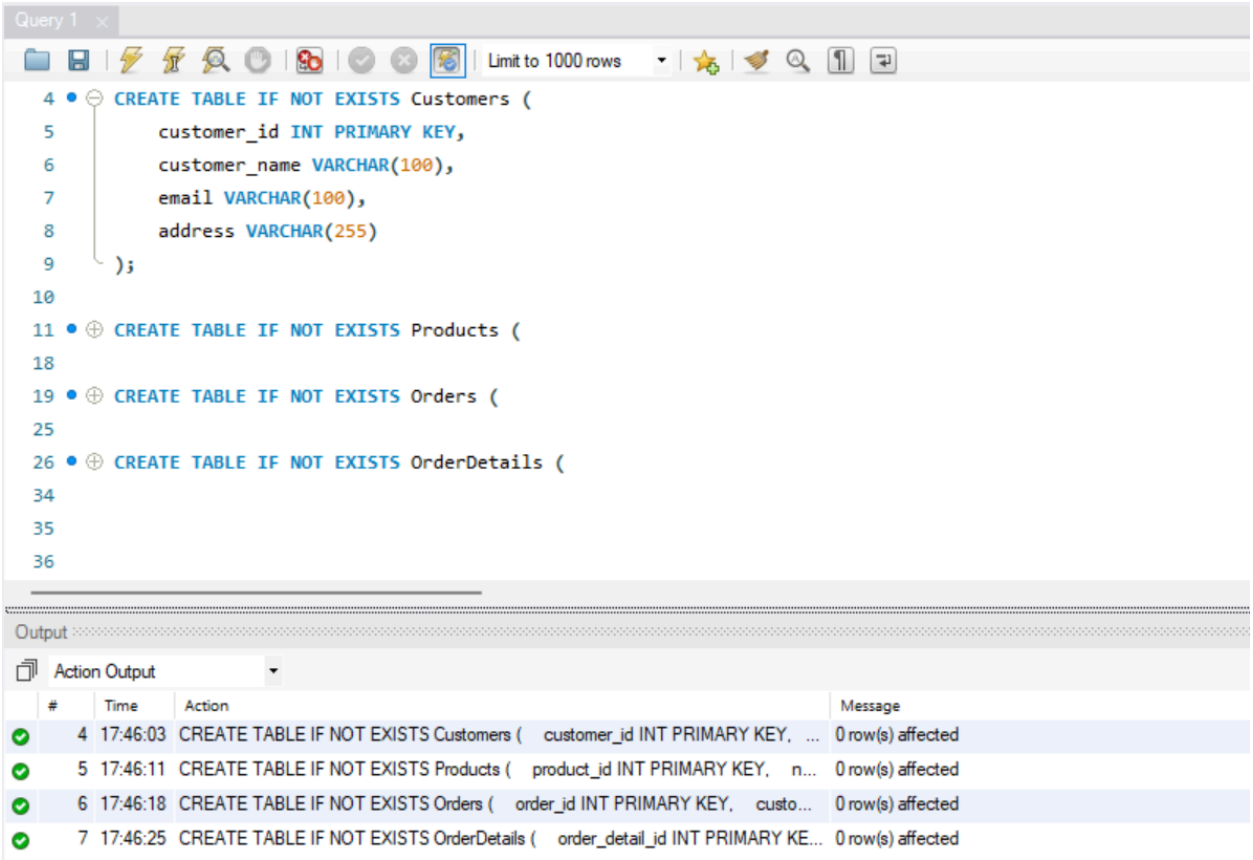
- Databases are essential for storing, managing, and retrieving structured data efficiently.
- They provide a systematic way of organizing and managing large amounts of data, ensuring data integrity, security, and easy accessibility.
- Without databases, managing data in applications would be error-prone, inefficient, and insecure.
-
- Example Scenario: A small e-commerce website needs to store information about its products, customers, orders, and payments.
- Without a database, managing this information using simple text files or spreadsheets would be chaotic, leading to data inconsistency, loss, and difficulties in retrieval.

Sample Data

1. **Create the Database** (if it doesn't exist):
2. **Select the Database**:



3. **Create the Tables** (if they don't exist):



4. Insert the Sample Data

INSERT INTO Customers (customer_id, customer_name, email, address)
VALUES
(1, 'John Doe', 'john@example.com', '123 Elm St'),
(2, 'Jane Smith', 'jane@example.com', '456 Oak St'),
(3, 'Alice Johnson', 'alice@example.com', '789 Pine St');

INSERT INTO Products (product_id, name, category, price, description)
VALUES
(1, 'Widget', 'Electronics', 25.99, 'A useful widget'),
(2, 'Gadget', 'Electronics', 55.99, 'An amazing gadget'),
(3, 'Thingamajig', 'Toys', 12.99, 'A fun toy'),
(4, 'Book', 'Books', 15.99, 'An interesting book');

INSERT INTO Orders (order_id, customer_id, order_date)
VALUES
(1, 1, '2024-01-15'),
(2, 2, '2024-02-20'),
(3, 3, '2024-03-25');

INSERT INTO OrderDetails (order_detail_id, order_id, product_id, quantity)
VALUES
(1, 1, 1, 2),
(2, 1, 2, 1),
(3, 2, 3, 4),
(4, 3, 4, 1),
(5, 3, 1, 1);

2. Basic Queries:
- Select
Retrieve all records from the Products table.
Example:
SELECT * FROM Products;

```
63 •  SELECT * FROM Products;
64
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	category	price	description
1	Widget	Electronics	25.99	A useful widget
2	Gadget	Electronics	55.99	An amazing gadget
3	Thingamajig	Toys	12.99	A fun toy
4	Book	Books	15.99	An interesting book
NULL	NULL	NULL	NULL	NULL

- Distinct
Retrieve unique product categories from the Products table.
Example:
SELECT DISTINCT category FROM Products;

```
65 •  SELECT DISTINCT category FROM Products;
66
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category
Electronics
Toys
Books

- Where

Retrieve all products priced above $50.

Example:

SELECT * FROM Products WHERE price > 50;

```
67 •   SELECT * FROM Products WHERE price > 50;
68
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	category	price	description
2	Gadget	Electronics	55.99	An amazing gadget
NULL	NULL	NULL	NULL	NULL

- And & Or

Retrieve products that are either in category 'Electronics' or priced above $100.

Example:

SELECT * FROM Products WHERE category = 'Electronics' OR price > 100;

```
69 •   SELECT * FROM Products WHERE category = 'Electronics' OR price > 100;
70
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	category	price	description
1	Widget	Electronics	25.99	A useful widget
2	Gadget	Electronics	55.99	An amazing gadget
NULL	NULL	NULL	NULL	NULL

- Order By

Retrieve all products, sorted by price in descending order.

Example:

SELECT * FROM Products ORDER BY price DESC;

```
71 •   SELECT * FROM Products ORDER BY price DESC;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	category	price	description
1	Widget	Electronics	25.99	A useful widget
2	Gadget	Electronics	55.99	An amazing gadget
NULL	NULL	NULL	NULL	NULL

- Insert Into

Add a new product to the Products table.

Example:

INSERT INTO Products (product_id, name, category, price, description)
VALUES
(5, 'Laptop', 'Electronics', 999.99, 'A high-performance laptop'),
(6, 'Smartphone', 'Electronics', 699.99, 'A latest model smartphone'),
(7, 'Board Game', 'Toys', 29.99, 'A fun family board game'),
(8, 'Novel', 'Books', 19.99, 'A thrilling new novel');

- Update
Update the price of a product.
Example:
UPDATE Products SET price = 19.99 WHERE name = 'Board Game';

✓	5	21:56:05	UPDATE Products SET price = 19.99 WHERE name = 'Board Game'	1 row(s) affecte

- Delete
Remove a product from the Products table.
Example:
DELETE FROM Products WHERE name = 'Book';

✓	6	21:57:37	UPDATE Products SET price = 19.99 WHERE name = 'Board Game'	0 row(s) affected Rows matched: 1

- Select Top
Retrieve the top 5 most expensive products.
Example:
SELECT TOP 5 * FROM Products ORDER BY price DESC;

```
85 •   SELECT * FROM Products ORDER BY price DESC LIMIT 5;
86
```

Result Grid | Filter Rows: | Edit: | Export/Import:

product_id	name	category	price	description
5	Laptop	Electronics	999.99	A high-performance laptop
6	Smartphone	Electronics	699.99	A latest model smartphone
2	Gadget	Electronics	55.99	An amazing gadget
1	Widget	Electronics	25.99	A useful widget
7	Board Game	Toys	19.99	A fun family board game
NULL	NULL	NULL	NULL	NULL

- Like

Retrieve products with names starting with 'B'.

Example:

SELECT * FROM Products WHERE name LIKE 'B%';

```
87 •   SELECT * FROM Products WHERE name LIKE 'B%';
88
```

Result Grid | Filter Rows: | Edit: | Export/Import:

product_id	name	category	price	description
4	Book	Books	15.99	An interesting book
7	Board Game	Toys	19.99	A fun family board game
NULL	NULL	NULL	NULL	NULL

- Wildcards

Retrieve products with names containing 'phone'.

Example:

SELECT * FROM Products WHERE name LIKE '%get%';

```
89 •   SELECT * FROM Products WHERE name LIKE '%get%';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

product_id	name	category	price	description
1	Widget	Electronics	25.99	A useful widget
2	Gadget	Electronics	55.99	An amazing gadget
NULL	NULL	NULL	NULL	NULL

- In

Retrieve products in specific categories.

Example:

SELECT * FROM Products WHERE category IN ('Electronics', 'Furniture');

```
91 •   SELECT * FROM Products WHERE category IN ('Electronics', 'Books');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	name	category	price	description
1	Widget	Electronics	25.99	A useful widget
2	Gadget	Electronics	55.99	An amazing gadget
4	Book	Books	15.99	An interesting book
5	Laptop	Electronics	999.99	A high-performance laptop
6	Smartphone	Electronics	699.99	A latest model smartphone
8	Novel	Books	19.99	A thrilling new novel
NULL	NULL	NULL	NULL	NULL

- Between

Retrieve products priced between $20 and $100.

Example:

SELECT * FROM Products WHERE price BETWEEN 20 AND 100;

```
93 •   SELECT * FROM Products WHERE price BETWEEN 20 AND 100;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

product_id	name	category	price	description
1	Widget	Electronics	25.99	A useful widget
2	Gadget	Electronics	55.99	An amazing gadget
NULL	NULL	NULL	NULL	NULL

- Aliases

Assign an alias to a column in the results.

Example:

SELECT name AS product_name, price AS product_price FROM Products;

```
SELECT name AS product_name, price AS product_price FROM Products;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name	product_price
Widget	25.99
Gadget	55.99
Thingamajig	12.99
Book	15.99
Laptop	999.99
Smartphone	699.99
Board Game	19.99
Novel	19.99

-- Joins

- Inner Join

Retrieve orders along with customer information.

Example:

```
SELECT Orders.order_id, Customers.customer_name
FROM Orders
INNER JOIN Customers ON Orders.customer_id = Customers.customer_id;
```

```
SELECT Orders.order_id, Customers.customer_name
    FROM Orders
    INNER JOIN Customers ON Orders.customer_id = Customers.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_name
1	John Doe
2	Jane Smith
3	Alice Johnson

- Left Join

Retrieve all customers and their orders, including those without orders.

Example:

```
SELECT Customers.customer_name, Orders.order_id
FROM Customers
LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

```
SELECT Customers.customer_name, Orders.order_id
 FROM Customers
 LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_name
1	John Doe
2	Jane Smith
3	Alice Johnson

- Right Join

Retrieve all orders and their customer information, including those without customer details.

Example:

```
SELECT Customers.customer_name, Orders.order_id
FROM Customers
RIGHT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

```
SELECT Customers.customer_name, Orders.order_id
FROM Customers
RIGHT JOIN Orders ON Customers.customer_id = Orders.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	order_id
John Doe	1
Jane Smith	2
Alice Johnson	3

- Union

Combine results from two queries.

Example:

```
SELECT customer_name AS name FROM Customers
UNION
SELECT name FROM Products;
```

```
SELECT customer_name AS name FROM Customers
UNION
SELECT name FROM Products;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name
John Doe
Jane Smith
Alice Johnson
Widget
Gadget
Thingamajig
Book
Laptop
Smartphone
Board Game
Novel

- Create DB

Create a new database named ShopDB.

Example:

```
CREATE DATABASE ShopDB;
```

25	22:16:58	CREATE DATABASE ShopDB	1 row(s) affecte

- Create Table

Create a new table Customers.

Example:

```
CREATE TABLE NewShop (
    new_id INT PRIMARY KEY,
    new_name VARCHAR(100),
    email VARCHAR(100) UNIQUE
);
```

26	22:18:38	CREATE TABLE NewShop (	new_id INT PRIMARY KEY,	new_name VA...	0 row(s) affected

-- Constraints

- Not Null

Ensure a column cannot have NULL values.

Example:

```
CREATE TABLE NewOrder(
    order_id INT NOT NULL,
    order_date DATE NOT NULL
);
```

- Unique

Ensure all values in a column are unique.

Example:

```
CREATE TABLE Users (
    user_id INT PRIMARY KEY,
    username VARCHAR(50) UNIQUE
);
```

- Primary Key

Create a primary key for a table.

Example:

```
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2)
);
```

- Foreign Key

Create a foreign key to link two tables.

Example:

```
CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

- Check

Ensure values in a column meet a specified condition.

Example:

```
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY,
    age INT CHECK (age >= 18)
);
```

- Default

Set a default value for a column.
Example:

```
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2) DEFAULT 0.00
);
```

- Create Index
Create an index on the name column of the Products table.
Example:

```
CREATE INDEX idx_name ON Products(name);
```

- Drop
Drop the Products table.
Example:

```
DROP TABLE Products;
```

- Alter
Add a new column to the Products table.
Example:

```
ALTER TABLE Products ADD description VARCHAR(255);
```

- Auto Increment
Create a table with an auto-incrementing primary key.
Example:

```
CREATE TABLE Products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2)
);
```

- Views
Create a view to simplify complex queries.
Example:

```
CREATE VIEW ProductView AS
SELECT product_id, name, price FROM Products WHERE price > 50;
```

- Null Values
Retrieve records with NULL values in a column.
Example:

```
SELECT * FROM Orders WHERE customer_id IS NULL;
```

Group By
Group records by a specified column.

```
SELECT category, COUNT(*) AS num_products
FROM Products
GROUP BY category;
```

- Having

Filter groups based on a condition.

Example:

```
SELECT category, COUNT(*) AS num_products
FROM Products
GROUP BY category
HAVING COUNT(*) > 10;
```

-- Functions

- Aggregate Functions

Calculate the average price of products.

Example:

```
SELECT AVG(price) AS avg_price FROM Products;
```

- Null Functions

Replace NULL values with a default value.

Example:

```
SELECT COALESCE(NULL, 'Default Value');
```

3. Callable statement, Prepared Statement, stored procedure

- Stored Procedure

Stored Procedures are precompiled collections of SQL statements stored in the database that can be executed as needed.

Example:

```
-- Stored procedure to get a customer name by customer_id
CREATE PROCEDURE GetCustomerName(IN customerId INT, OUT customerName VARCHAR(100))
BEGIN
  SELECT customer_name INTO customerName FROM Customers WHERE customer_id = customerId;
END;
```

Usage:

```
-- Call the stored procedure
CALL GetCustomerName(1, @customerName);

-- Retrieve the output parameter value
SELECT @customerName;
```

2. Prepared Statement
Prepared Statements are used to execute parameterized SQL queries, allowing for dynamic input values while helping to prevent SQL injection.
Example:

```
-- Using prepared statements in MySQL
    PREPARE stmt FROM 'SELECT * FROM Customers WHERE customer_id = ?';
    SET @customerId = 1;
    EXECUTE stmt USING @customerId;
    DEALLOCATE PREPARE stmt;
```

3. Callable Statement
Callable Statements are used to call stored procedures from SQL.
Example:

```
-- First, create a stored procedure for adding a new customer
    CREATE PROCEDURE AddCustomer(IN customerName VARCHAR(100), IN email
VARCHAR(100), IN address VARCHAR(255))
    BEGIN
        INSERT INTO Customers (customer_name, email, address) VALUES (customerName, email,
address);
    END;
```

Usage:

```
-- Call the stored procedure to add a new customer
    CALL AddCustomer('Alice Johnson', 'alice.johnson@example.com', '789 Pine St');
```

4. Concept of normalization:
Each table now contains data specific to a single entity type, reducing redundancy and ensuring data integrity.

- Unnormalized Table:

OrderID	CustomerName	CustomerAddress	ProductName	Quantity
1	John Doe	123 Elm St	Widget	4
2	Jane Smith	456 Oak St	Gadget	2

- First Normal Form (1NF):

OrderID | CustomerName | CustomerAddress | ProductName | Quantity

---
1 | John Doe | 123 Elm St | Widget | 4
2 | Jane Smith | 456 Oak St | Gadget | 2

- Second Normal Form (2NF):

Orders Table:
OrderID | ProductName | Quantity | CustomerID
---
1 | Widget | 4 | 1
2 | Gadget | 2 | 2

Customers Table:
CustomerID | CustomerName | CustomerAddress
---
1 | John Doe | 123 Elm St
2 | Jane Smith | 456 Oak St

- Third Normal Form (3NF):

Orders Table:
OrderID | ProductID | Quantity | CustomerID
---
1 | 1 | 4 | 1
2 | 2 | 2 | 2

Products Table:
ProductID | ProductName
---
1 | Widget
2 | Gadget

Customers Table:
CustomerID | CustomerName | CustomerAddress
---
1 | John Doe | 123 Elm St
2 | Jane Smith | 456 Oak St