**GLAMWEB**

**GlamWeb: Your Ultimate Fashion Destination**

Welcome to **GlamWeb**, where style meets elegance! We are your one-stop shop for all things fashion, offering a curated selection of high-quality clothing, accessories, and beauty products that cater to the modern, sophisticated woman.

**What We Offer**

* **Indian Wear**

Discover the beauty of traditional attire with our exquisite collection of sarees, lehengas, and kurtas. Perfect for any festive occasion or cultural celebration, our Indian wear line blends timeless elegance with contemporary designs.

* **Footwear**

Step out in style with our chic and comfortable footwear. From everyday sandals to glamorous heels, our collection ensures you have the perfect pair for every outfit and occasion.

* **Western Wear**

Elevate your wardrobe with our range of western wear. Whether you're looking for casual jeans and tops, elegant dresses, or professional workwear, GlamWeb has you covered with the latest trends and timeless classics.

* **Jewellery**

Add a touch of sparkle to your look with our stunning jewellery collection. From statement necklaces and earrings to delicate rings and bracelets, our pieces are designed to complement any outfit and make you shine.

* **Makeup**

Unleash your inner beauty with our premium makeup products. Featuring everything from vibrant lipsticks to flawless foundations, our makeup line helps you create any look, from natural to bold, with ease and confidence.

**Our Vision**

At GlamWeb, we believe that fashion is a powerful form of self-expression. Our mission is to empower women to feel confident and beautiful in their own skin by providing stylish, high-quality products that enhance their unique sense of style.

SQL:

DBMS

RBMS

**IMPLEMENTATION**

**ER DIAGRAM**

An Entity-Relationship (ER) diagram is a visual representation of the data model for a system. It illustrates how entities (such as objects, people, or concepts) relate to each other within that system.

ER diagrams are used primarily in database design and are a key tool for understanding and organizing the structure of a database. They help in mapping out the relationships and interactions between different entities, ensuring the database is well-organized and can support the necessary queries and operations.

The main components of ER model are: entity set and relationship set.Here are the geometric shapes and their meaning in an ER Diagram

**Entities:**

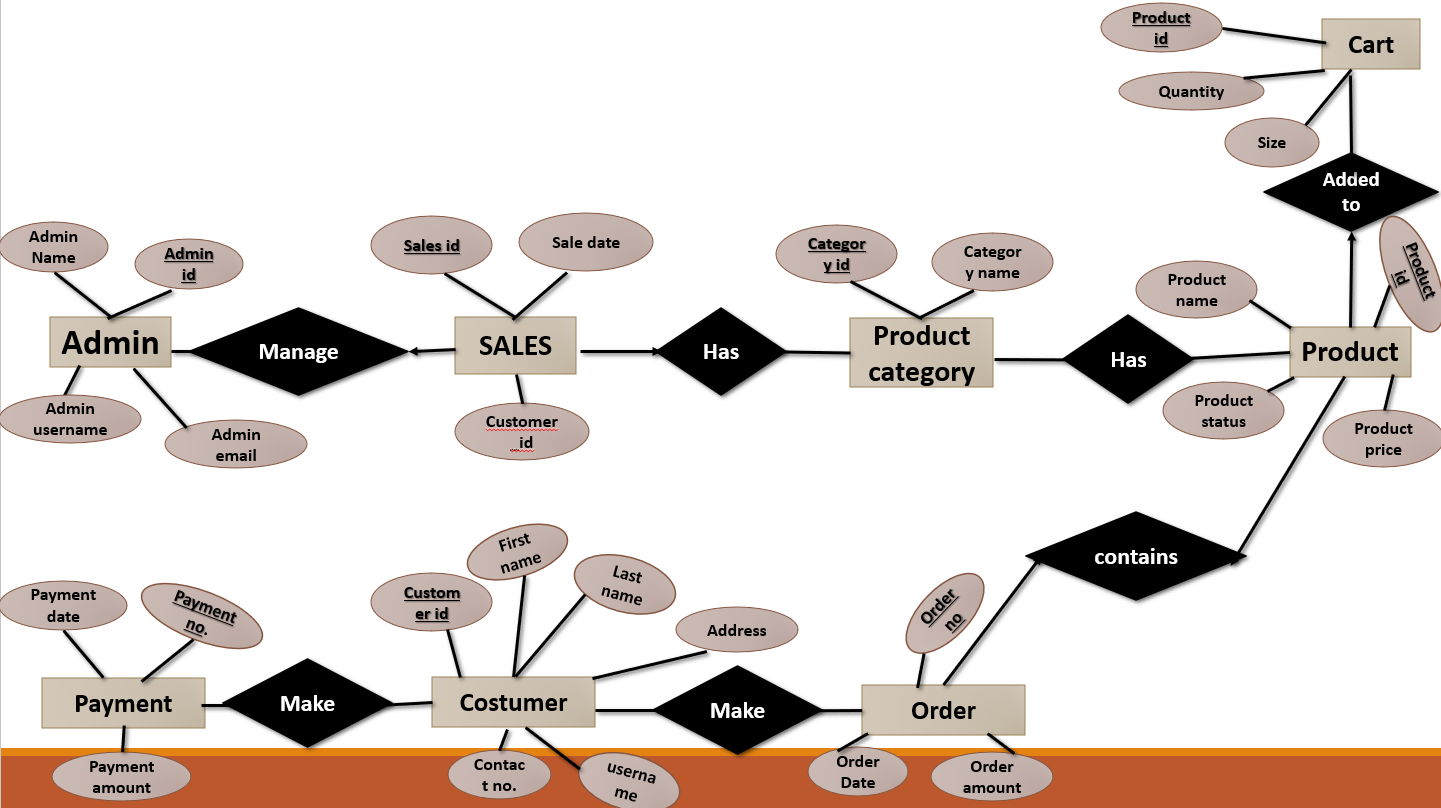
* **Definition:** Entities are objects or things in the system that have a distinct existence. Each entity represents a set of similar items that are of interest in the database.
* **Representation:** Entities are depicted by rectangles.

**Attributes:**

* **Definition:** Attributes are properties or characteristics of an entity.
* **Representation:** Attributes are shown by ovals connected to their respective entities.

**Relationships:**

* **Definition:** Relationships depict how entities interact with each other. They describe associations between entities.
* **Representation:** Relationships are represented by diamonds, with lines connecting them to the entities involved.



**NORMALIZATION**

Normalization is a systematic approach to organizing a database to reduce redundancy and improve data integrity. The process involves decomposing a database into smaller, related tables without losing data. The primary objectives of normalization are to eliminate duplicate data, ensure data dependencies make sense, and protect the data from anomalies such as insertion, update, and deletion anomalies.

**Normal Forms**

Normalization is carried out in stages, known as normal forms (NF). Each stage ensures that certain criteria are met to structure the database in a particular way.

* **First Normal Form (1NF):**
* Each column in a table must contain only atomic (indivisible) values.
* Each row in a table must be unique.
* All entries in a column must be of the same data type.
* **Second Normal Form (2NF):**
* The table must be in First Normal Form (1NF).
* All non-key attributes must be fully functionally dependent on the entire primary key, meaning there should be no partial dependencies on a part of a composite primary key.

In other words, 2NF ensures that each non-key attribute is dependent on the whole primary key, not just a part of it.

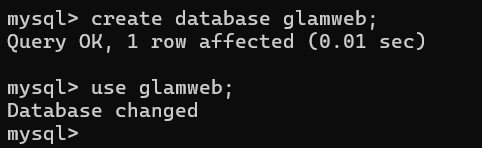
* **Third Normal Form (3NF)**:
* The table must be in Second Normal Form (2NF).
* All non-key attributes must be directly dependent on the primary key, and not on other non-key attributes (i.e., there should be no transitive dependencies).
* **Boyce-Codd Normal Form (BCNF) requires that:**
* The table must be in Third Normal Form (3NF).
* Every determinant in the table must be a candidate key.

**SQL QUERIES**

**1.create database:**

create database glamweb;

use glamweb;



**2.create table admin:**

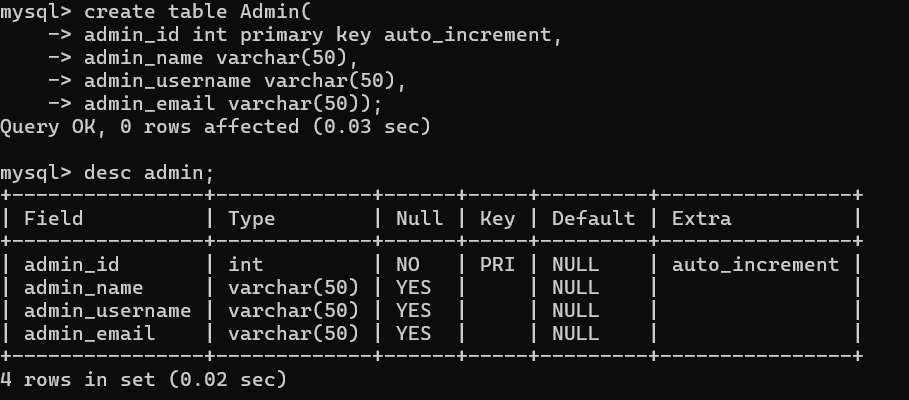
create table Admin(

-> admin\_id int primary key auto\_increment,

-> admin\_name varchar(50),

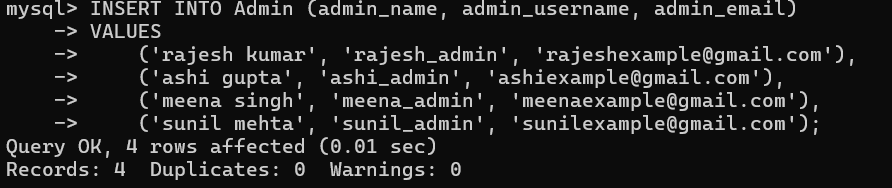
-> admin\_username varchar(50),

-> admin\_email varchar(50));

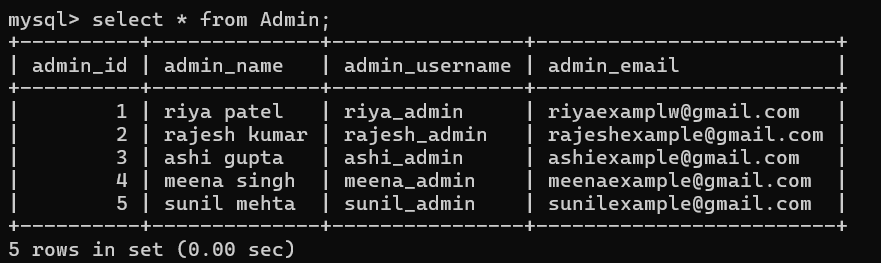


3.Insert values in Admin table:

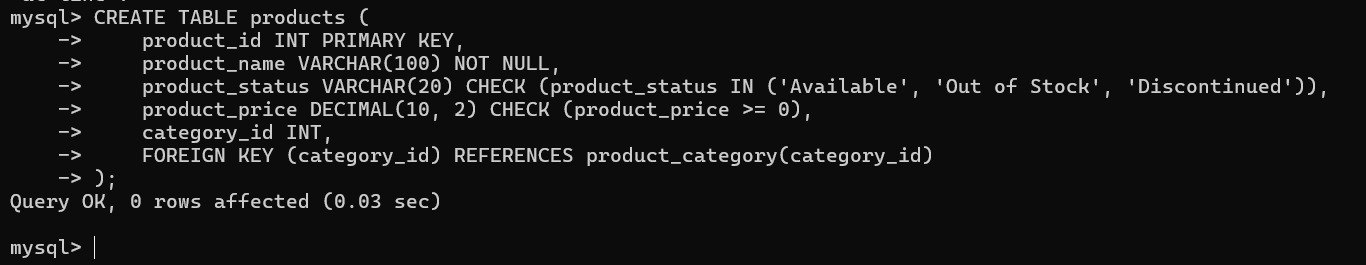




**OUTPUT:**



4.create table of products:

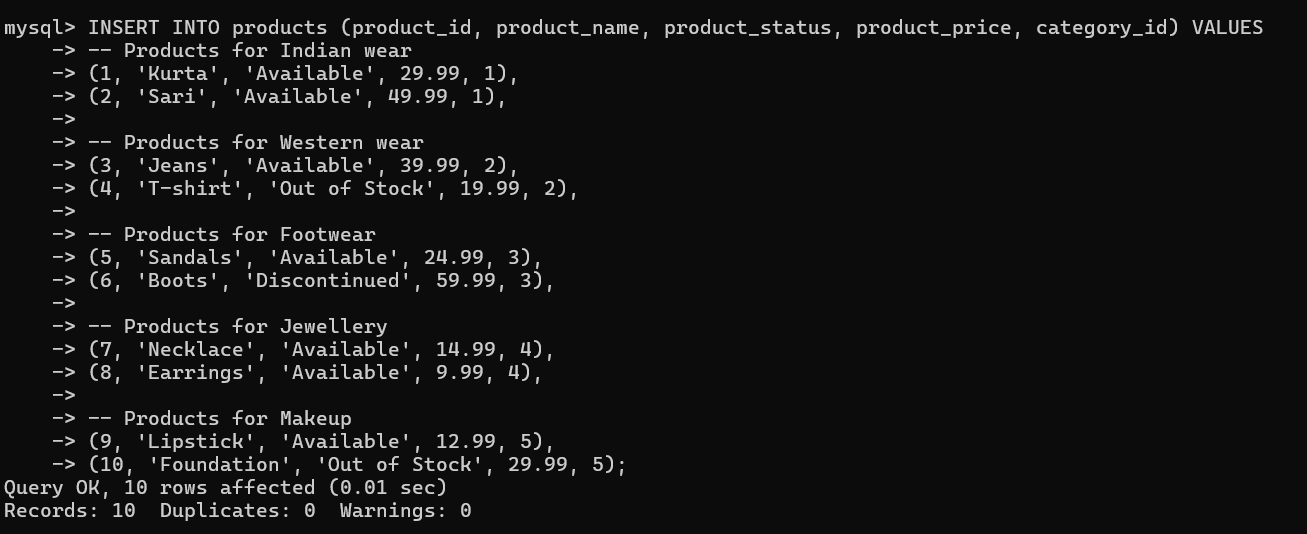


* **CONSTRAINTS:**

This products table includes:

* product\_id as the primary key.
* product\_name with a NOT NULL constraint, meaning it must have a value.
* product\_status with a CHECK constraint to ensure the value is either 'Available', 'Out of Stock', or 'Discontinued'.
* product\_price with a CHECK constraint to ensure the price is non-negative.
* category\_id as a foreign key that references the category\_id in the product\_category table.

**5.insert into products table:**



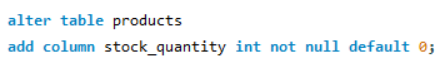
**OUTPUT:**



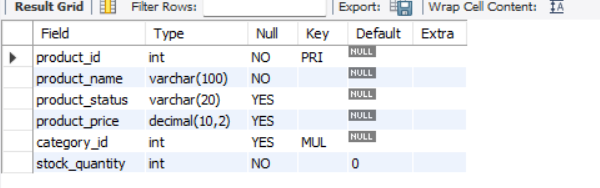
**ALTER QUERIES**

**1.ADD A NEW COLUMN**

add a new column to track the product's stock quantity.



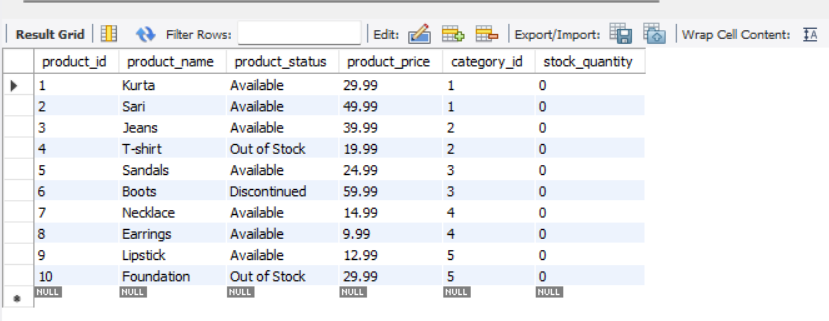
**OUTPUT:**



**2.Modify an existing column**



**OUTPUT:**



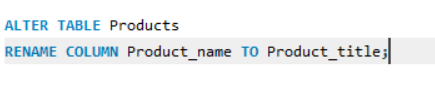
**3.Add a new CONSTRAINT**



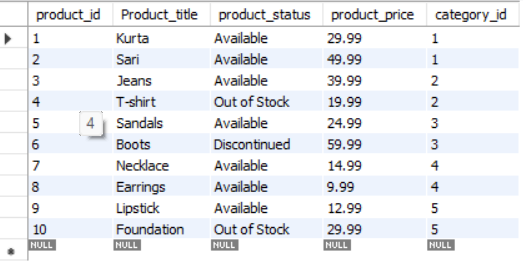
**4.Drop column**



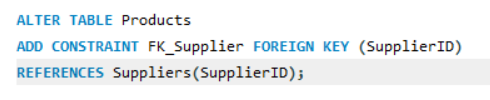
**5.Rename a column**

****

**OUTPUT:**

****

**6.** **Add a Foreign Key Constraint**

****

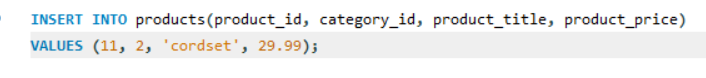
**7.TRUNCATE**

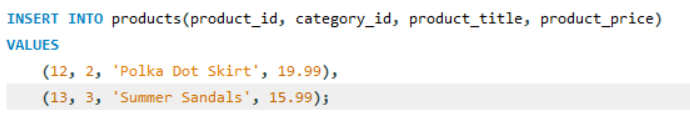
****

**[DML] DATA MANIPULATION LANGUAGE**

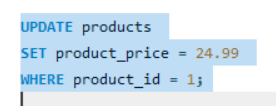
The Full Form of DML is Data Manipulation Language. Data Manipulation Language (DML) is a programming language, also a category of SQL, applying useful table format and formulas for exchanging information, with the data limited to a point in the database, using SQL.

**1. Insert Queries**





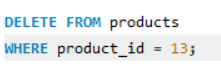
**2. Update Queries**

****

**OUTPUT:**

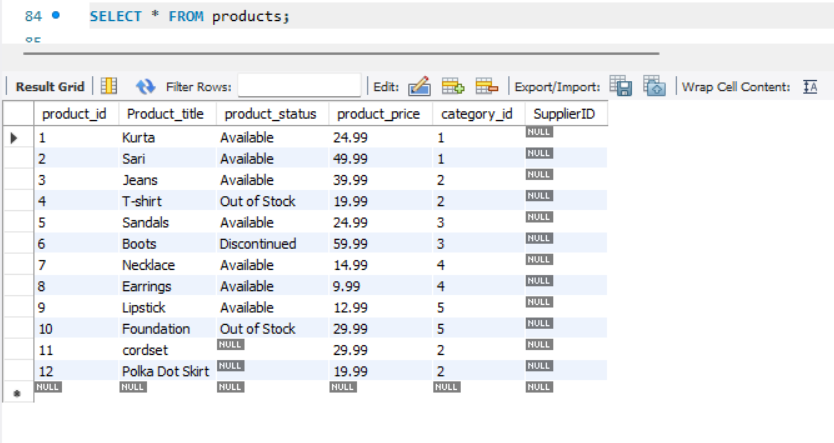
****

**4. Delete Queries**

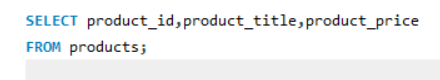
****

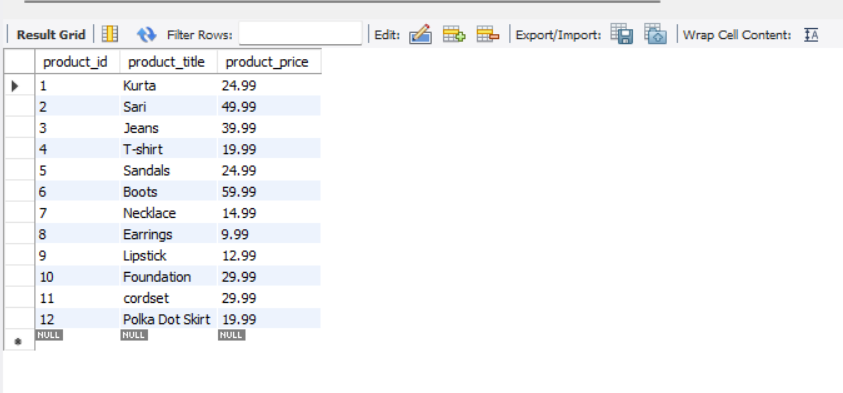
**5.Select statements:**

* **Select All Products**

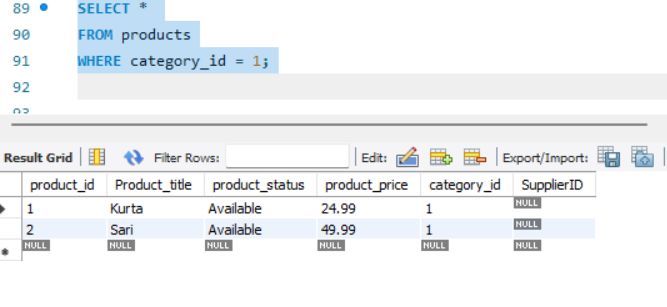
****

* **Select Specific Columns**

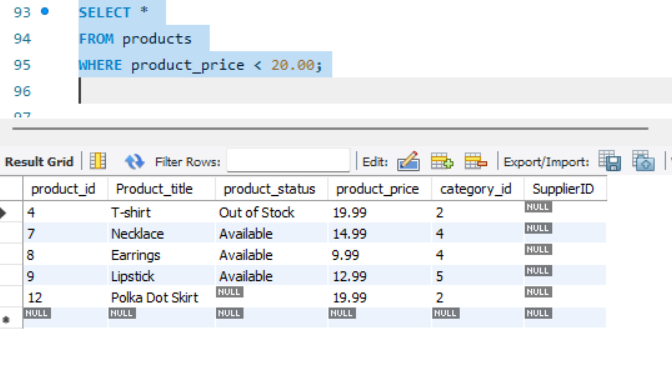
****



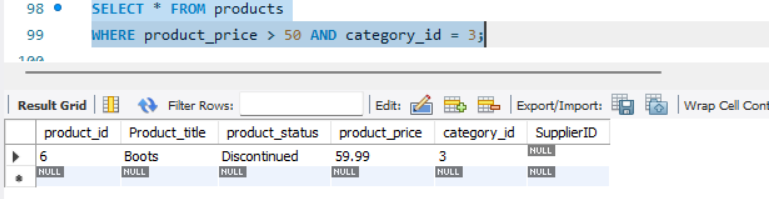
* **Select Products by Category**



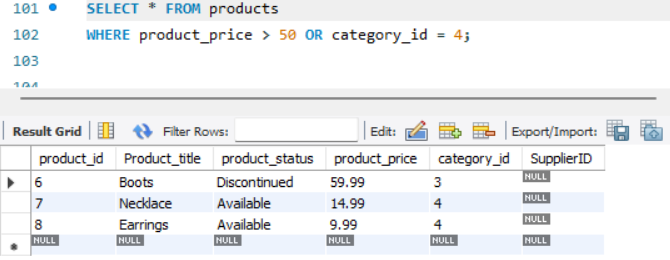
**3. Select Rows with a Condition**

****

* **Select Rows with Multiple Conditions**

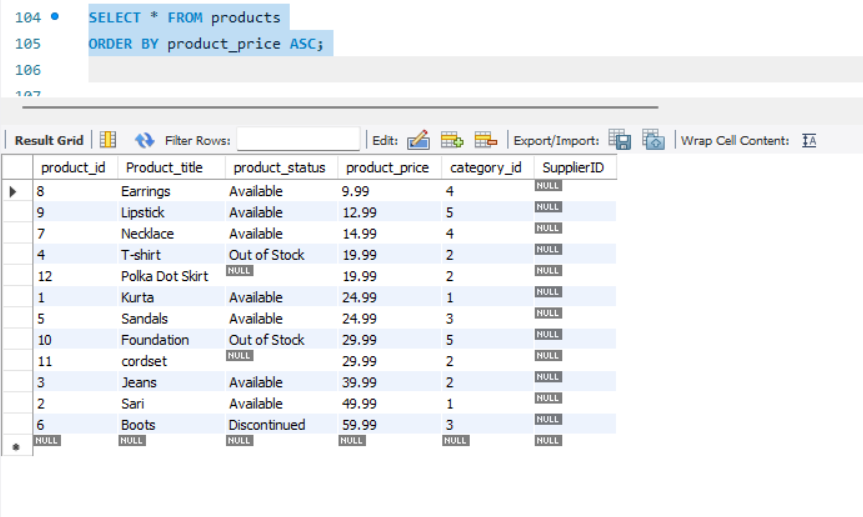


* **Select statement by using ‘or’ funtion**

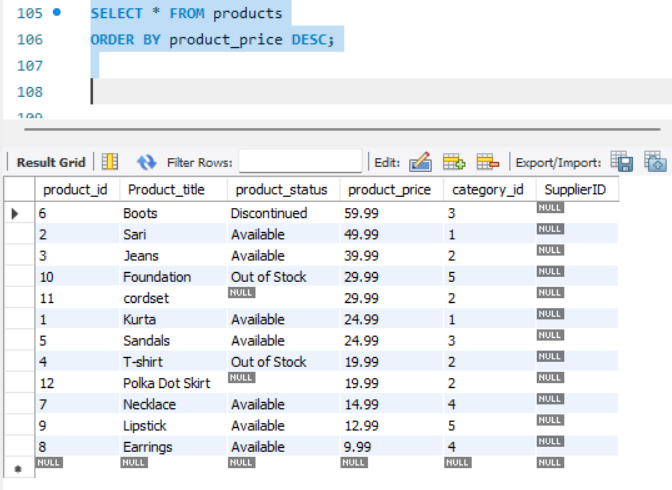
****

* **Select statement using order by ascending and descending:**

**ASCENDING:**

****

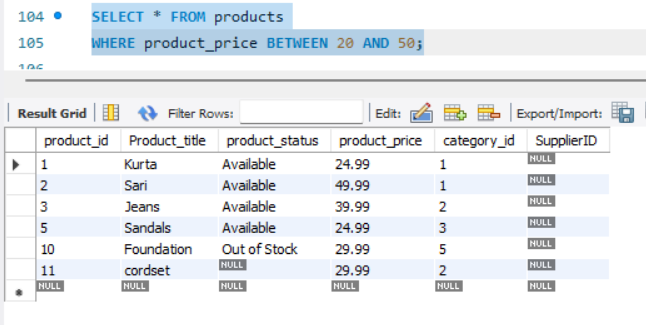
**DESCENDING:**

****

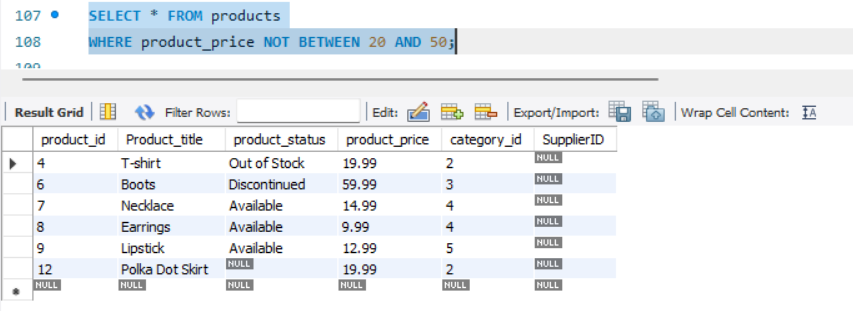
* **Select record by using range:**

**To select records using a range in SQL, you typically use the BETWEEN operator, which allows you to specify a range of values for filtering**

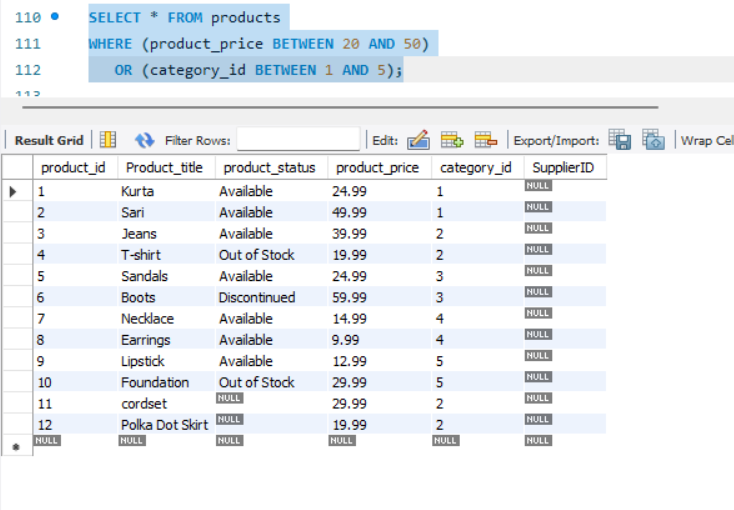
**Retrieve products where the price is between 20 and 50:**

****

**Retrieve products where the price is not between 20 and 50:**

****

**Retrieve products where the price is between 20 and 50, or the category\_id is between 1 and 5:**

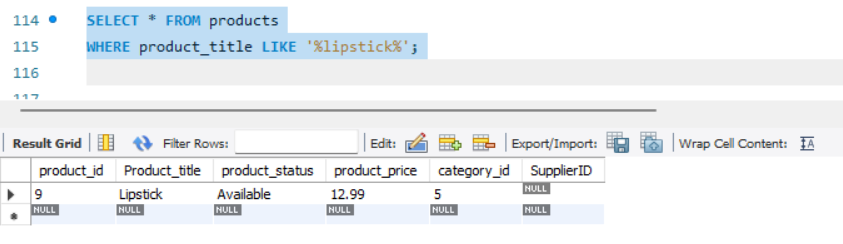
****

* **Select statement using LIKE operator:**

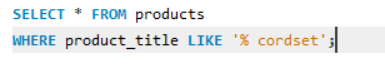
**The LIKE operator in SQL is used for pattern matching with text data. It allows you to filter results based on whether a column's value matches a specified pattern.**

**Wildcard Characters**

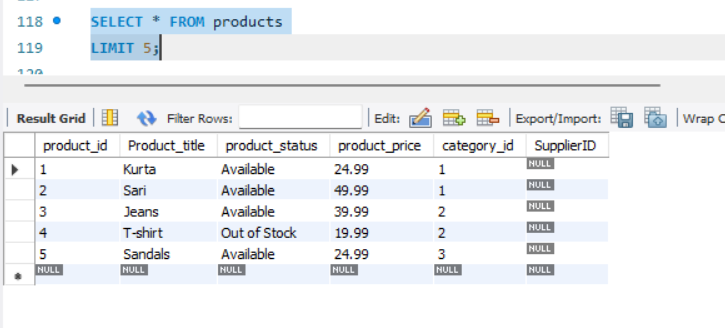
* **%: Represents zero or more characters.**
* **\_: Represents a single character.**
* **Select records if only one word nis known for specific column.**
* **Searching for 'lipstick' Anywhere in the Column**

****

* **To select products where the last word in the product\_name column is 'cordset', you can use:**

****

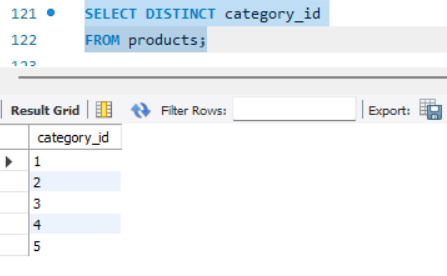
* **Select record using limit:**

****

* **select record using distinct.**

**Retrieve Unique Values from a Single Column**

**To get unique values from the category column in the products table:**

****

**AGGREGATE FUNCTIONS**

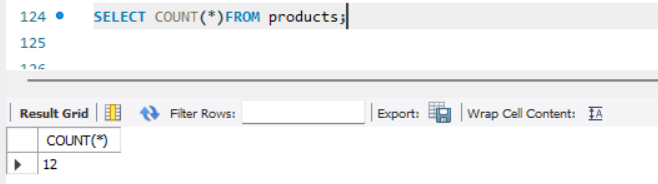
Aggregate functions in SQL are used to perform calculations on a set of values and return a single value. These functions are commonly used with the GROUP BY clause to group rows that have the same values into summary rows.

**Common Aggregate Functions**

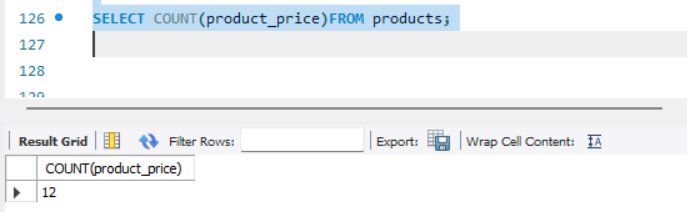
1. **COUNT**: Returns the number of rows in a set.
2. **SUM**: Returns the total sum of a numeric column.
3. **AVG**: Returns the average value of a numeric column.
4. **MIN**: Returns the minimum value in a set.
5. **MAX**: Returns the maximum value in a set.

* **Select statement by using count:**

Count all rows:

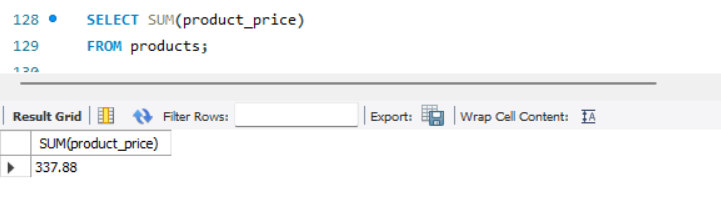


Count values in specific column:

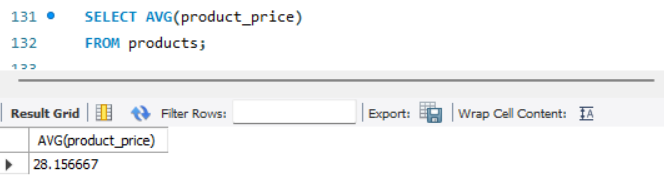


* **Select statement by using SUM:**

**Sum of the price column:**

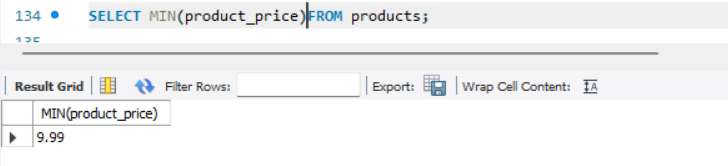
****

* **Select statement by using AVG:**

****

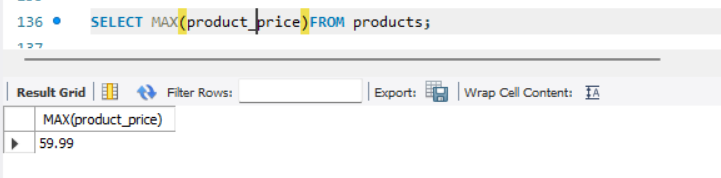
* **Select statement using MIN:**

Minimum price of products:



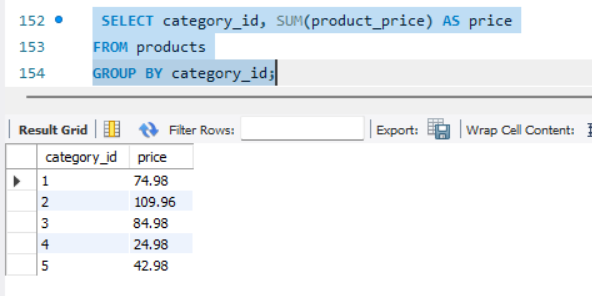
* **Select statement using MAX:**

Maximum price of products:

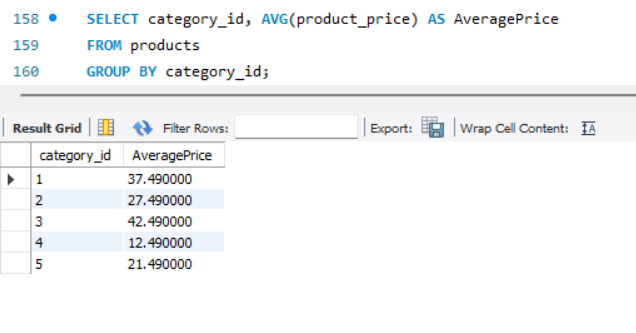


**AGGREGATE FUNCTIONS WITH GROUP BY CLAUSE:**

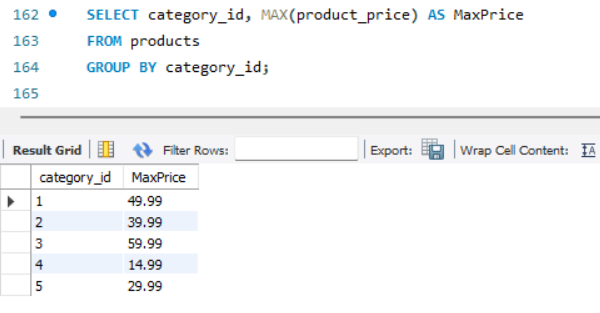
**SUM**



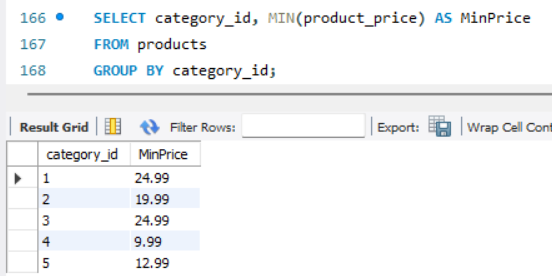
**AVERAGE**



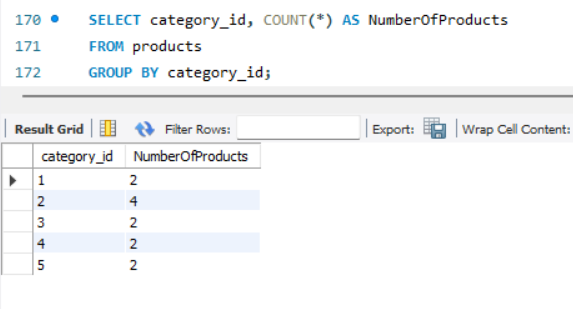
**MAX**



**MIN**



**COUNT**



**STRING**

In programming and databases, a **string** is a sequence of characters, typically used to represent text. Strings can include letters, numbers, symbols, and spaces. In most programming languages and SQL databases, strings are enclosed in single quotes (' ') or double quotes (" ").

**Common String Functions:**

Here’s an overview of common string functions used in SQL and most programming languages:

**1.CONCAT()**:

* + **Definition**: Concatenates (joins together) two or more strings into one.

**2.LOWER()**:

* **Definition**: Converts all characters in a string to lowercase.

**3.UPPER()**:

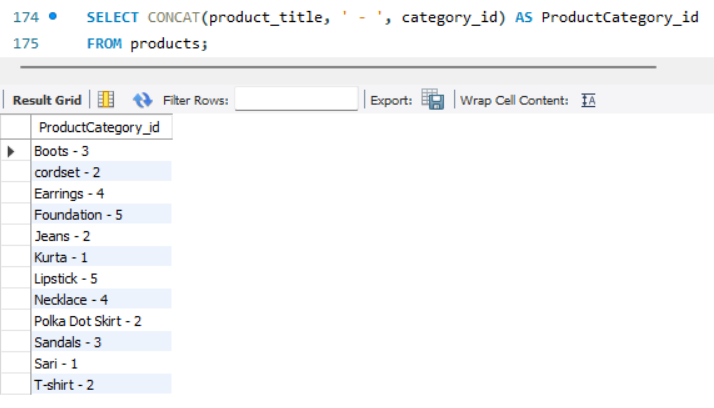
* **Definition**: Converts all characters in a string to uppercase.

**4.LENGTH()** (or LEN() in some databases):

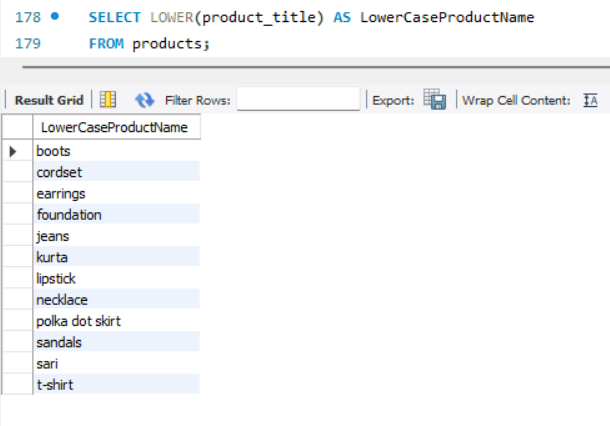
* **Definition**: Returns the number of characters in a string.

**5.SUBSTRING()** (or SUBSTR()):

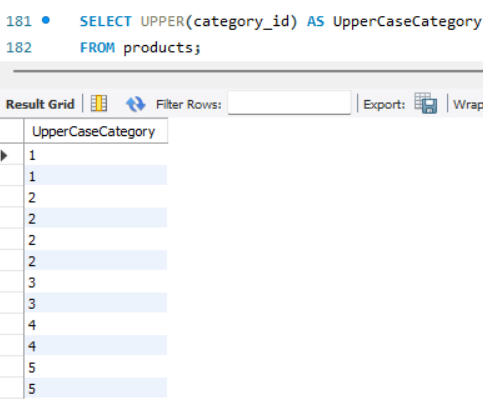
* **Definition**: Extracts a substring from a string, starting at a specified position and for a specified length.
* **CONCAT**



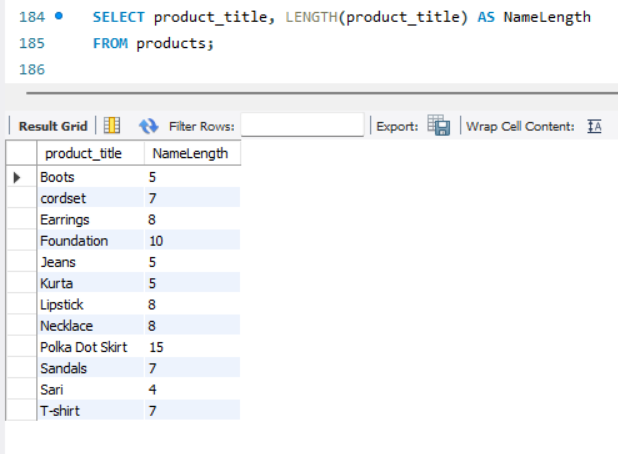
* **LOWER**



* **UPPER**



* **LENGTH**



**COMPARISON FUNCTION**

**1. Greater Than (>):**

* **Definition**: Selects rows where a column's value is greater than the specified value.

**2. Smaller Than (<):**

* **Definition**: Selects rows where a column's value is smaller than the specified value.

**3. Equal To (=):**

* **Definition**: Selects rows where a column's value is equal to the specified value.

**4. Not Equal To (<> or !=):**

* **Definition**: Selects rows where a column's value is not equal to the specified value.

**5. Greater Than or Equal To (>=):**

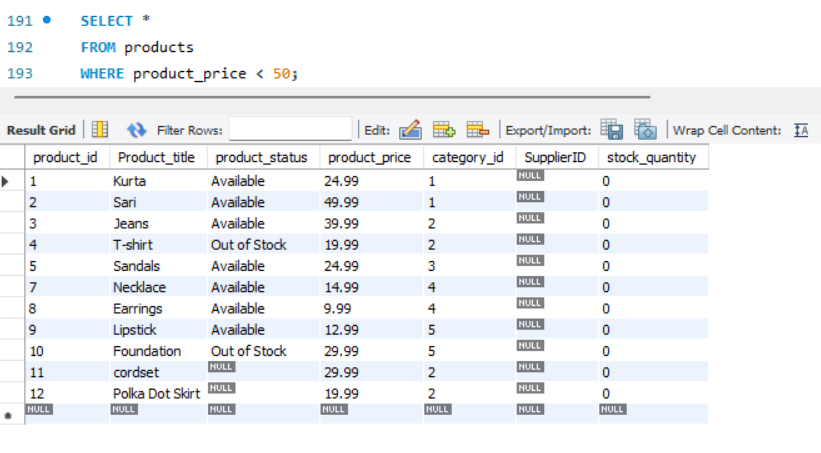
**Definition**: Selects rows where a column's value is greater than or equal to the specified value.

**6. Smaller Than or Equal To (<=):**

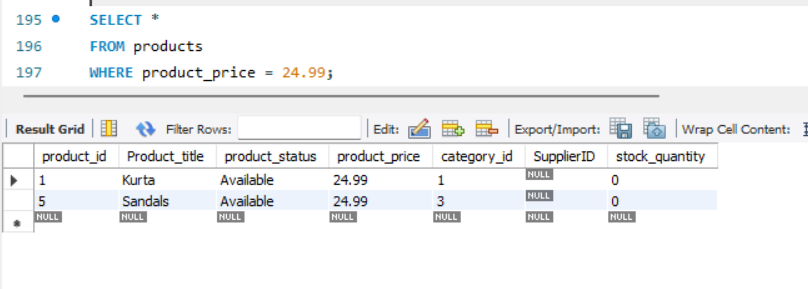
* **Definition**: Selects rows where a column's value is smaller than or equal to the specified value.
* **Greater Than (>):**

****

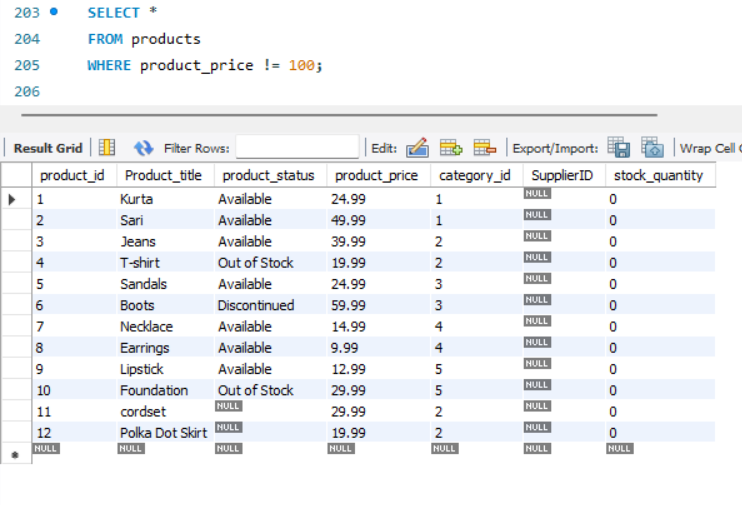
* **Smaller Than (<):**

****

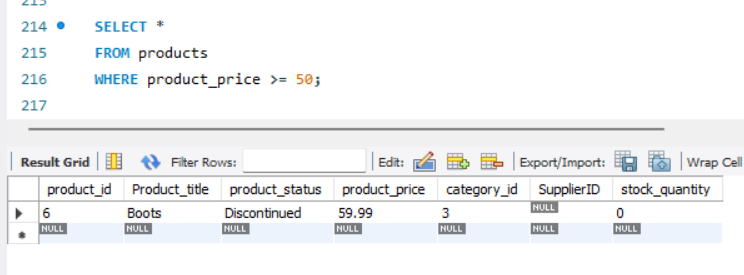
* **Equal To (=):**



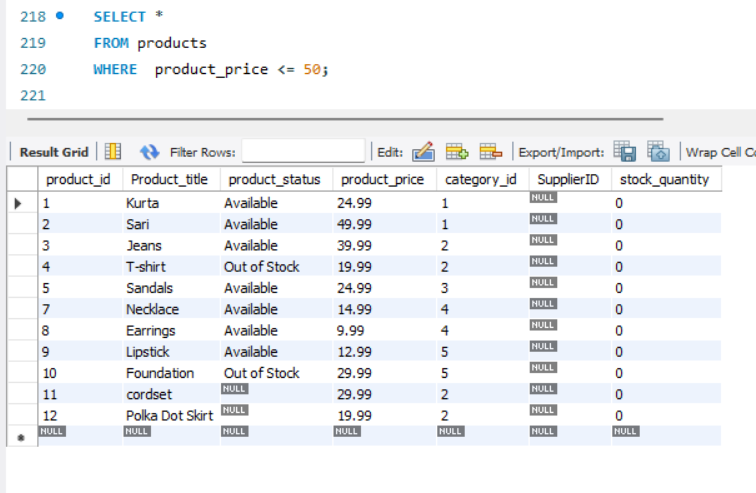
* **Not Equal To (<> or !=):**

****

* **Greater Than or Equal To (>=)**:

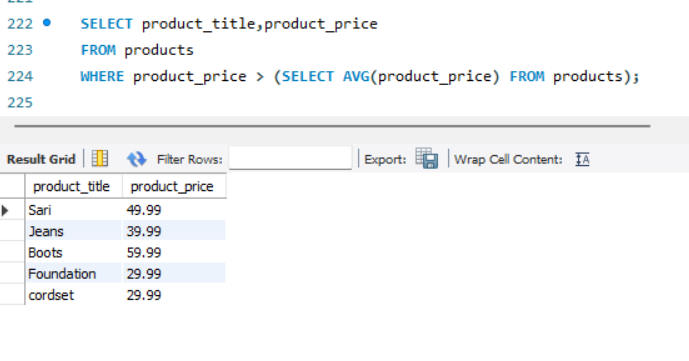


* **Smaller Than or Equal To (<=):**

****

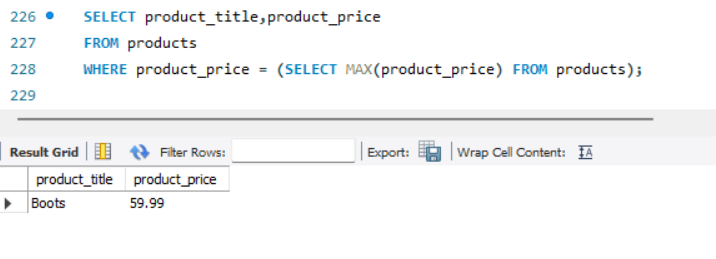
**SUBQUERY**

**Definition: A query embedded within another query. It can be used in various parts of a SQL query, such as the SELECT, FROM, or WHERE clauses.**

****

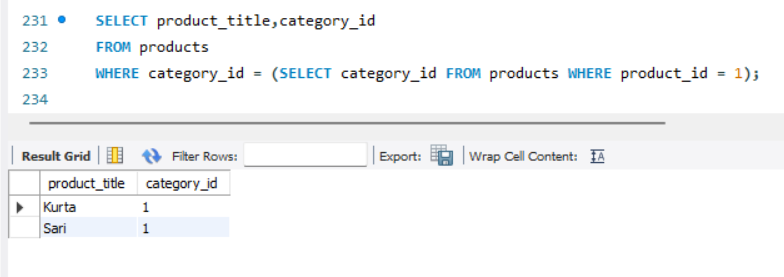
**SINGLE ROW SUBQUERY:**

Definition: A subquery that returns a single row and a single column. It's often used with comparison operators like =, >, <.

****

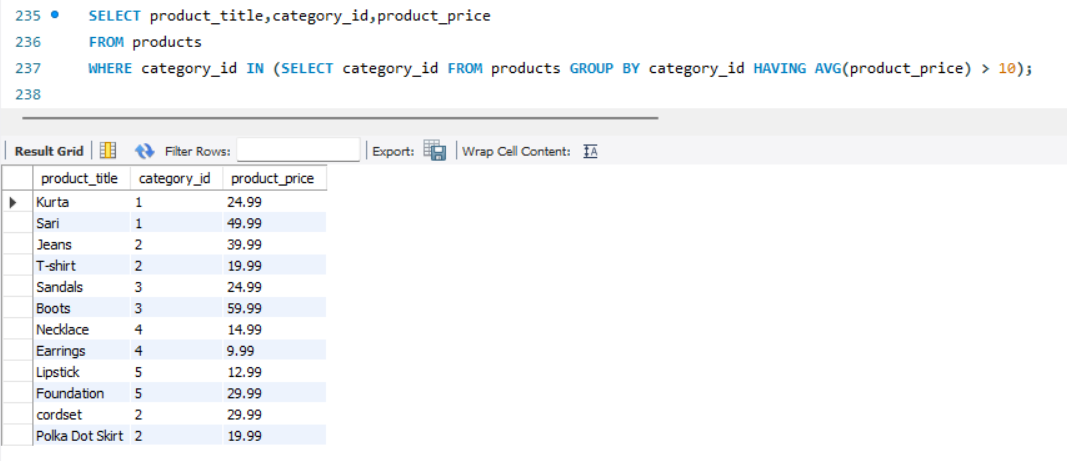
**Multiple-Row Subquery:**

* **Definition: A subquery that returns multiple rows, often used with operators like IN, ANY, or ALL.**

****

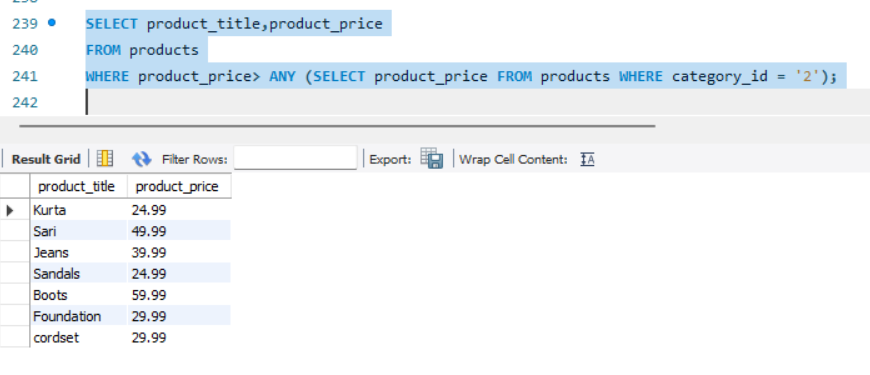
**IN Operator:**

* **Definition: Used to filter rows based on whether a column's value matches any value in a list or a subquery result.**

****

**ANY Operator:**

* **Definition: Compares a value to each value returned by the subquery. If any comparison returns TRUE, the overall result is TRUE.**

****

**ALL Operator:**

* **Definition: Compares a value to every value returned by the subquery. The condition must hold true for all rows returned by the subquery.**

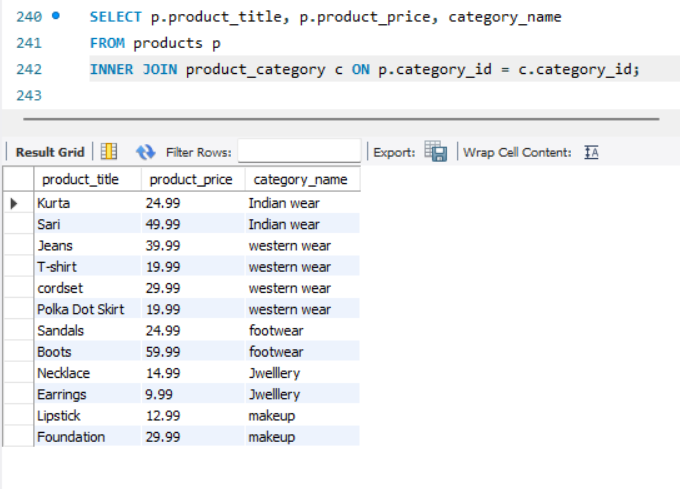
****

**JOINS**

**Joins are used in SQL to combine rows from two or more tables based on a related column between them. Below, I’ll describe the different types of joins, with examples on how they can be applied using a products and product\_category table.**

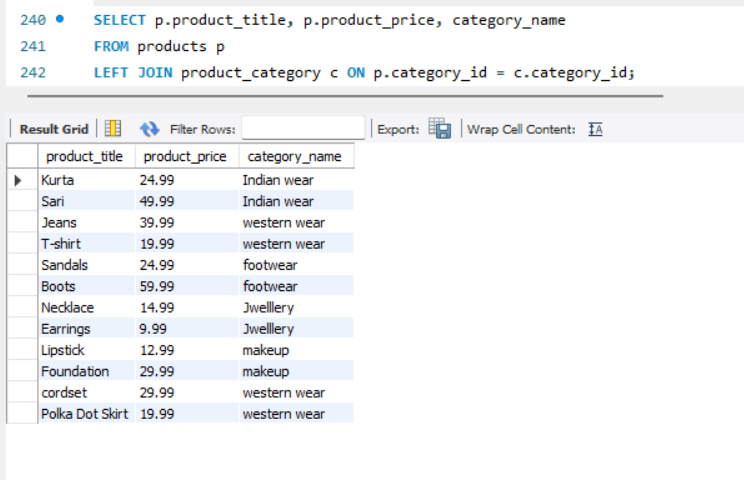
**1.INNER JOIN**

* **Definition: Returns only the rows that have matching values in both tables.**

****

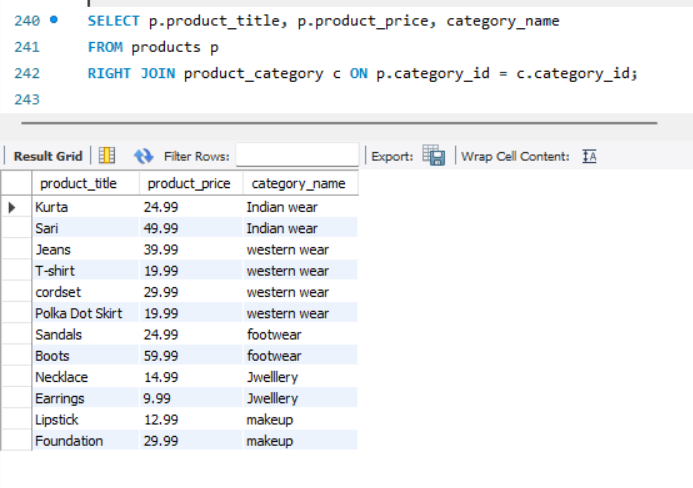
**2. LEFT JOIN (or LEFT OUTER JOIN)**

* **Definition: Returns all rows from the left table, and the matched rows from the right table. If no match is found, NULL values are returned for columns from the right table**

****

**3. RIGHT JOIN (or RIGHT OUTER JOIN)**

* **Definition: Returns all rows from the right table, and the matched rows from the left table. If no match is found, NULL values are returned for columns from the left table.**

****

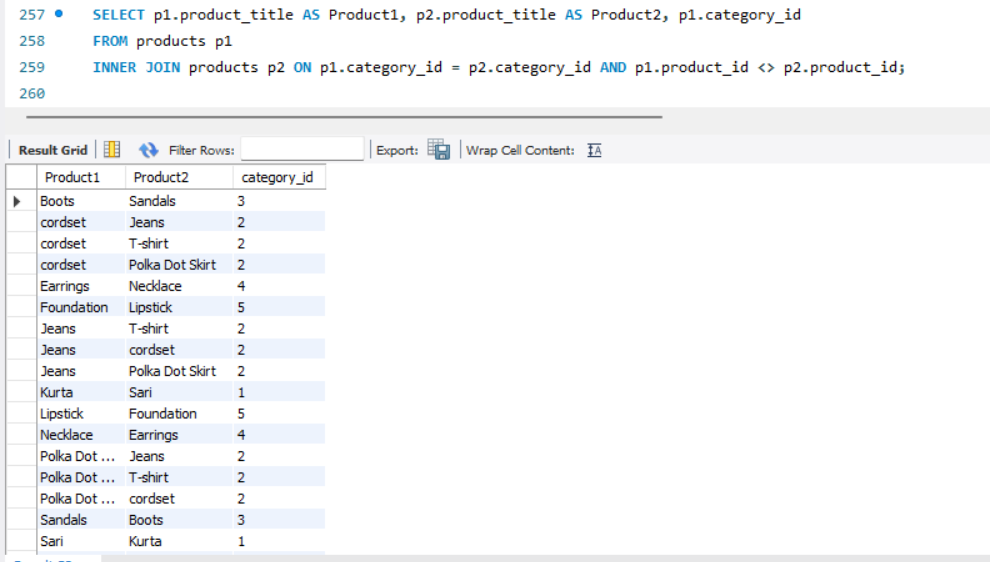
**4. FULL OUTER JOIN**

* **Definition: Returns all rows when there is a match in either left or right table. Rows that do not have a match in either table will have NULLs for the non-matching columns.**

****

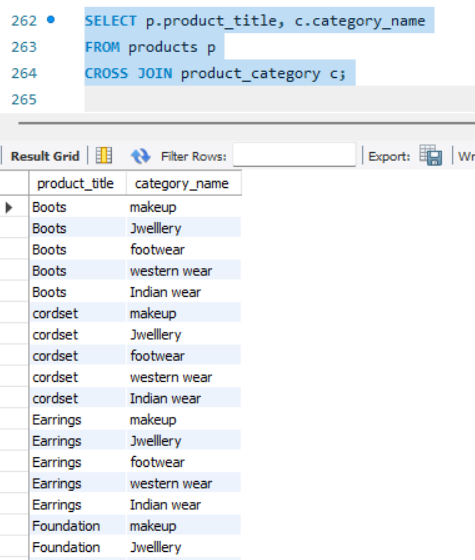
**5. SELF JOIN**

* **Definition: A self-join is a join of a table with itself.**

****

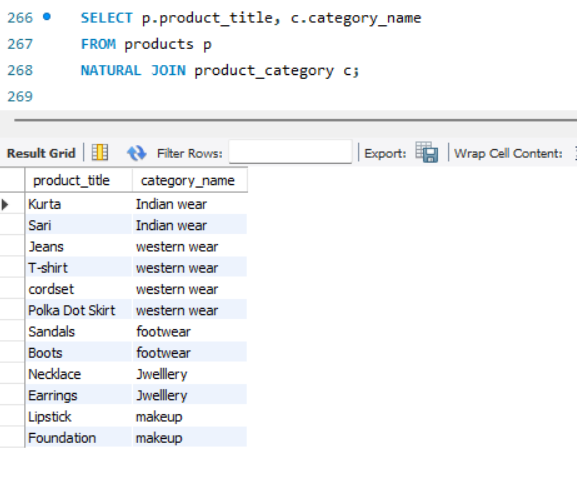
**6. CROSS JOIN**

* **Definition: Returns the Cartesian product of the two tables, meaning it combines each row from the first table with each row from the second table.**

****

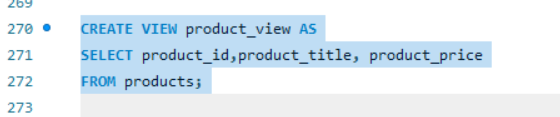
**7. NATURAL JOIN**

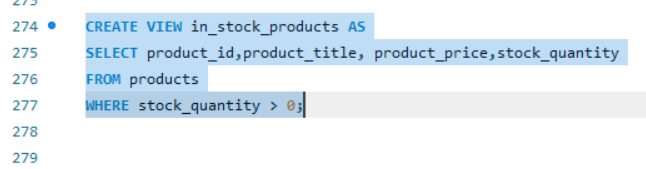
* **Definition: Automatically joins tables based on all columns with the same name in both tables.**

****

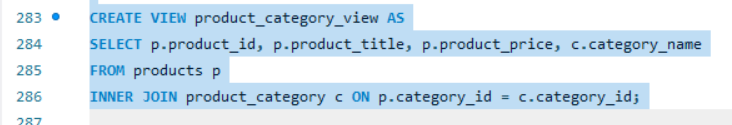
**VIEW**

**1. Simple View**

****

**2. View with Filtering**

**3. View with Joins**

****

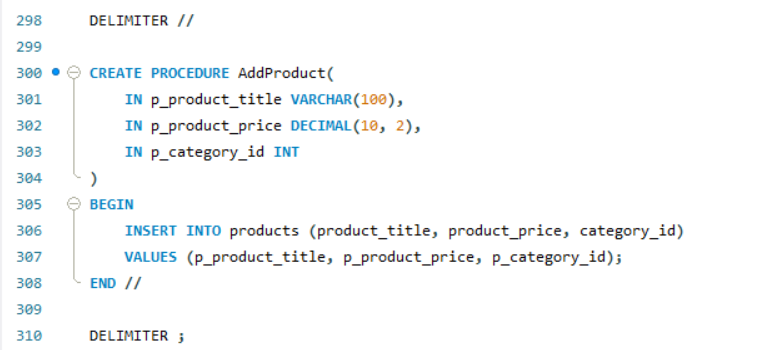
**4.Drop the Existing View**

****

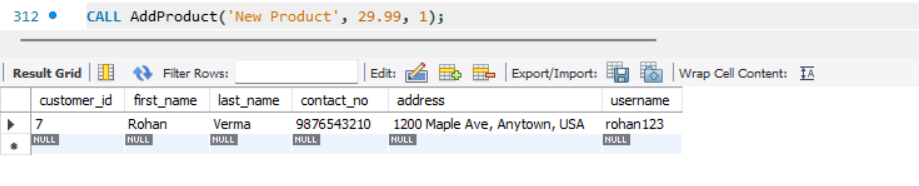
**STORE PROCEDURE**

**Stored procedures in MySQL are a way to encapsulate and reuse SQL queries. You can use stored procedures to perform various operations on the products table, such as inserting, updating, deleting, or retrieving data.**

**1.create store procedure:**

****

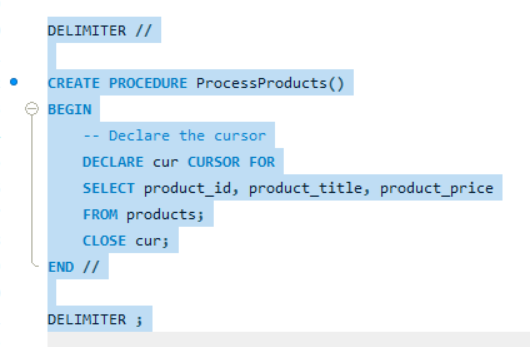
**Usage:**

****

**Explanation**

* **DELIMITER //: Changes the statement delimiter to // to define the procedure. This prevents issues with semicolons within the procedure definition.**
* **CREATE PROCEDURE: Defines a new stored procedure.**
* **IN: Specifies input parameters.**
* **BEGIN ... END: Encapsulates the procedure logic.**
* **CALL: Executes the stored procedure.**

**CURSOR**

****

* **Declare Cursor: Defines the cursor with the DECLARE cursor\_name CURSOR FOR SELECT ... statement.**
* **Open Cursor: Initializes the cursor with OPEN cursor\_name.**
* **Fetch Rows: Retrieves data row-by-row with FETCH cursor\_name INTO ....**
* **Handle End of Cursor: Manages the end of the result set with a CONTINUE HANDLER.**
* **Close Cursor: Frees resources with CLOSE cursor\_name.**

**TRIGGER**

Triggers are special types of stored procedures in SQL that are automatically

executed (or &quot;triggered&quot;) in response to certain events on a particular table or

view. These events typically include INSERT, UPDATE, or DELETE

operations.

