-- This SQL query retrieves distinct ProductName and UnitPrice combinations from the Products table, selecting the top 20 most expensive products based on their UnitPrice, and orders the results by UnitPrice in descending order.

SELECT DISTINCT ProductName as Twenty\_Most\_Expensive\_Products, UnitPrice -- Selects distinct ProductName and UnitPrice combinations from the Products table, aliasing ProductName as Twenty\_Most\_Expensive\_Products

FROM Products AS a -- Specifies the table from which to retrieve data, aliasing it as 'a'

WHERE 20 >= (SELECT COUNT(DISTINCT UnitPrice) -- Filters the rows to include only the top 20 most expensive products based on their UnitPrice

FROM Products AS b -- Subquery: Specifies the table from which to retrieve data, aliasing it as 'b'

WHERE b.UnitPrice>= a.UnitPrice) -- Subquery: Counts the number of distinct UnitPrice values greater than or equal to the UnitPrice of the current row in table 'a'

ORDER BY UnitPricedesc; -- Orders the result set by the UnitPrice column in descending order

SELECT p.ProductID, p.Product, SUM(s.Amt \* s.Price) AS total

FROM Sells AS s

LEFT JOIN Product AS p ON s.ProductID = p.ProductID

WHERE s.Day=2 GROUP BY s.Day, s.Product

SELECT \* FROM `posts` WHERE `ad\_price` >= 2000 AND `ad\_price` <= 5000

>= 2000 AND `ad\_price` <= 5000

$query = "SELECT \* FROM `posts` WHERE `ad\_title` LIKE CONCAT('%',?,'%') AND `ad\_brand`

LIKE CONCAT('%',?,'%') AND `ad\_price` >= ? AND `ad\_price` <= ?";

$get\_posts = $conn\_posts->prepare($query);

$get\_posts->bind\_param("ssss", $title, $brand, $min\_range, $max\_range);

-- This query calculates the average value of the 'pro\_price' column for each distinct value of 'pro\_com'.

SELECT AVG(pro\_price), pro\_com

-- Specifies the table from which to retrieve the data (in this case, 'item\_mast').

FROM item\_mast

-- Groups the result set by the 'pro\_com' column.

GROUP BY pro\_com;

select combo\_of\_item,count(combo\_of\_item) Count\_of\_Cust from (

select Cust\_ID ,string\_agg(Item\_ID,',') combo\_of\_item from (

select distinct \* from [table] ) a

group by Cust\_ID) b

group by Combo\_of\_item

SELECT Column1, Column2, ... FROM Table ORDER BY Column1 [ASC|DESC], Column2 [ASC|DESC], ...

SELECT count(\*) AS counter

FROM reviews

WHERE customers\_id = 12345

SELECT customers\_review

FROM reviews

WHERE

products\_id = '170'

-- Selecting columns 'pro\_name' as "Product Name", 'pro\_price' as "Price," and 'com\_name' as "Company"

SELECT P.pro\_name AS "Product Name",

P.pro\_price AS "Price",

C.com\_name AS "Company"

-- Specifying the tables to retrieve data from ('item\_mast' aliased as 'P' and 'company\_mast' aliased as 'C') and defining the relationship between them

FROM item\_mast P, company\_mast C

-- Filtering the results based on the condition that 'pro\_com' in 'P' matches 'com\_id' in 'C'

WHERE P.pro\_com = C.com\_id

-- Further filtering the results based on the condition that 'pro\_price' in 'P' is equal to the maximum 'pro\_price' returned by a subquery

AND P.pro\_price =

-- Subquery: Selecting the maximum 'pro\_price' from 'item\_mast' (aliased as 'P') where 'pro\_com' matches 'com\_id' in 'C'

(

SELECT MAX(P.pro\_price)

FROM item\_mast P

WHERE P.pro\_com = C.com\_id

);

DECLARE @colors table

(color varchar(55))

INSERT INTO @colors

SELECT DISTINCT color FROM saleslt.Product

SELECT p.productID, p.name, p.color

FROM saleslt.product AS p

WHERE color=@color;

SELECT Product\_t.Product\_ID, Product\_t.Product\_Name, SUM(Uses\_t.Footage \* Raw\_Materials\_t.Unit\_price ) AS Total\_Cost\_Of\_Raw\_Materials

FROM Product\_t, Uses\_t, Raw\_materials\_t

WHERE Product\_t.Product\_ID=Uses\_t.Product\_ID and Uses\_t.Material\_ID=Raw\_materials\_t.Material\_ID

GROUP BY Product\_ID

ORDER BY SUM(Raw\_Materials\_t.Unit\_price \* Uses\_t.Footage) DESC;

mysql> SELECT \*FROM employees ORDER BY Joining\_Date DESC;

SELECT COUNT(DISTINCT Country) ASDistinctCountries FROM customers

SELECT Contries.CountryName, Count (\*) AS Number

FROM Users

INNER JOIN Countries

ON Users.CountryId = Countries.CountryId

GROUP BY Countries.CountryName

ORDER BY Countries.CountryName

SELECT lastname AS ln

FROM

(SELECT lastname, count(\*) as Counter

FROM `students`

GROUP BY `lastname`) AS tbl WHERE Counter > 2

$minPrice = $request['min'];

$maxPrice = $request['max'];

$colors = $request['color'];

$sizes = $request['size'];

if (count($request->all()) != 0) {

$query = Product::with(['sizes' => function($query) use($request) {

$sizeArray = $request->get('size');

$query->whereIn('size', $sizeArray);

}]);

if(isset($minPrice) && isset($maxPrice)) {

$query->whereBetween('price', array($minPrice, $maxPrice));

}

if(isset($colors)) {

$query->whereIn('color\_id', $colors);

}

$products = $query->get();

}

-- Selecting the average 'pro\_price' as "Average Price" and 'com\_name' from the 'company\_mast' table as "Company"

SELECT AVG(pro\_price) AS "Average Price",

company\_mast.com\_name AS "Company"

-- Specifying the tables to retrieve data from ('item\_mast' and 'company\_mast') and defining the relationship between them

FROM item\_mast, company\_mast

-- Filtering the results based on the condition that 'pro\_com' in 'item\_mast' matches 'com\_id' in 'company\_mast'

WHERE item\_mast.pro\_com = company\_mast.com\_id

-- Grouping the results by 'com\_name' from 'company\_mast'

GROUP BY company\_mast.com\_name

-- Filtering the grouped results based on the condition that the average 'pro\_price' is greater than or equal to 350

HAVING AVG(pro\_price) >= 350;

SELECT p.ProductName

, c.CategoryName

From Products p

Join Categories c

ON p.ProductID = c.CategoryID

SELECT Products.ProductName, Categories.CategoryName

FROM Products

JOIN Categories ON Categories.CategoryID = Products.CategoryID

SELECT p.ProductName, c.CategoryName

FROM Products p

JOIN Categories c ON p.CategoryID = c.CategoryID

Select the names of all the products in the store.

select Name from Products;

Select the names and the prices of all the products in the store.

select name, price from products;

Select the name of the products with a price less than or equal to $200.

select name from products where price <= 200;

Select all the products with a price between $60 and $120.

select \* from products where price between 60 and 120;

select \* from products where price >= 60 and price <= 120;

Select the name and price in cents (i.e., the price must be multiplied by 100).

select name, price\*100 from products;

select name, concat(price\*100, ' cents') from products;

Compute the average price of all the products.

select avg(price) from products;

select sum(price)/count(price) from products;

WITH Part1Customers AS (

SELECT TABLE1.Customer, TABLE1.[Month]

FROM TABLE1

WHERE EXISTS (

SELECT 1

FROM TABLE2

WHERE TABLE1.MATERIAL = TABLE2.MATERIAL

)

)

SELECT TABLE1.\*

FROM TABLE1

WHERE EXISTS (

SELECT 1

FROM Part1Customers

WHERE Part1Customers.Customer = TABLE1.Customer

AND Part1Customers.[Month] = TABLE1.[Month]

select products.productname, products.price,categories.categoryname from products,categories where products.categoryid=categories.id;

or

SELECT products.productname, products.price, categories.categoryname FROM products INNER JOIN categories ON products.categoryid=categories.id;

-- This query selects specific columns and renames them for better readability.

-- It renames the 'pro\_name' column to "Item Name" and the 'pro\_price' column to "Price in Rs."

SELECT pro\_name as "Item Name", pro\_price AS "Price in Rs."

-- Specifies the table from which to retrieve the data (in this case, 'item\_mast').

FROM item\_mast;

SELECT \*

FROM Customer AS C

INNER JOIN

Orders AS O

ON C.CustomerID = O.CustomerID

WHERE C.Country = 'Germany' AND O.ShipCountry = 'Tahiti';