PROJECT 2

Select Statement

1. Retrieve all columns from the `stolen_vehicles` table.

2. Select only the `vehicle_type`, `make_id`, and `color` columns from the `stolen_vehicles` table.

From Statement

1. Write a query to display all records from the `make_details` table.

2. Retrieve all columns from the `locations` table.

Where Statement

1. Find all stolen vehicles that are of type "Trailer".

2. Retrieve all stolen vehicles that were stolen after January 1, 2022.

3. Find all stolen vehicles that are of color "Silver".

Group By and Order By

1. Count the number of stolen vehicles for each `vehicle_type` and order the results by the count in descending order.

2. Find the total number of stolen vehicles for each `make_id` and order the results by `make_id`.

Using Having vs. Where Statement

1. Find the `make_id` values that have more than 10 stolen vehicles.

2. Retrieve the `vehicle_type` values that have at least 5 stolen vehicles.

Limit and Aliasing

1. Retrieve the first 10 records from the `stolen_vehicles` table and alias the `vehicle_type` column as "Type".

2. Find the top 5 most common colors of stolen vehicles and alias the count column as "Total".

Joins in MySQL

1. Join the `stolen_vehicles` table with the `make_details` table to display the `vehicle_type`, `make_name`, and `color` of each stolen vehicle.

2. Join the `stolen_vehicles` table with the `locations` table to display the `vehicle_type`, `region`, and `country` where the vehicle was stolen.

Unions in MySQL

1. Write a query to combine the `make_name` from the `make_details` table and the `region` from the `locations` table into a single column.

PROJECT 2

2. Combine the `vehicle_type` from the `stolen_vehicles` table and the `make_type` from the `make_details` table into a single column.

Case Statements

1. Create a new column called "Vehicle_Category" that categorizes vehicles as "Luxury" if the `make_type` is "Luxury" and "Standard" otherwise.

2. Use a CASE statement to categorize stolen vehicles as "Old" if the `model_year` is before 2010, "Mid" if between 2010 and 2019, and "New" if 2020 or later.

Aggregate Functions

1. Calculate the total number of stolen vehicles.

2. Find the average population of regions where vehicles were stolen.

3. Determine the maximum and minimum `model_year` of stolen vehicles.

String Functions

1. Retrieve the `make_name` from the `make_details` table and convert it to uppercase.

2. Find the length of the `vehicle_desc` for each stolen vehicle.

3. Concatenate the `vehicle_type` and `color` columns from the `stolen_vehicles` table into a single column called "Description".

Update Records

1. Update the `color` of all stolen vehicles with `vehicle_type` "Trailer" to "Black".

2. Change the `make_name` of `make_id` 623 to "New Make Name" in the `make_details` table.

Bonus Questions

1. Write a query to find the top 3 regions with the highest number of stolen vehicles.

2. Retrieve the `make_name` and the total number of stolen vehicles for each make, but only for makes that have more than 5 stolen vehicles.

3. Use a JOIN to find the `region` and `country` where the most vehicles were stolen.

4. Write a query to find the percentage of stolen vehicles that are of type "Boat Trailer".

5. Use a CASE statement to create a new column called "Density_Category" that categorizes regions as "High Density" if `density` is greater than 500, "Medium Density" if between 200 and 500, and "Low Density" if less than 200.