Model Paper

A retail company is developing an **Inventory Management System** to track its products, suppliers, and stock movements. The database consists of three main tables.The **Products** table stores information about each product, including a unique `product_id` which is the primary key, `product_name` as a string that cannot be null, `price` as a decimal value which must be greater than zero, and `added_on` which records the date the product was added, defaulting to the current date.The **Suppliers** table keeps details about suppliers, with a unique `supplier_id` as the primary key, the supplier's full name stored in `supplier_name`, `contact_email` which must be unique to avoid duplicates, and `established_year` which must be a year not earlier than 2000.The **StockMovements** table tracks the movement of products in and out of the warehouse. It includes a unique `movement_id` as the primary key, foreign keys `product_id` referencing the `Products` table and `supplier_id` referencing the `Suppliers` table, a `movement_type` column which can only be 'IN' or 'OUT', and a `quantity` column that must be a positive integer.The system should also support stored procedures to retrieve product stock levels and use indexes to improve query performance.

1.

Write a SQL statement to create the `Products` table by applying all necessary constraints as described in the case study. Ensure the primary key is correctly defined, the product name is not null, the price has a check constraint to ensure it is greater than zero, and the added date defaults to the current date.

2.

Write a SQL statement to create the `Suppliers` table applying all required constraints mentioned in the case study. Make sure the primary key is set, the contact email is unique, and the established year has a check constraint to ensure it is no earlier than 2000.

3.

Write a SQL statement to create the `StockMovements` table with all necessary constraints as outlined in the case study. Ensure the primary key is set, foreign keys reference the `Products` and `Suppliers` tables properly, the movement type accepts only the values 'IN' or 'OUT', and the quantity is a positive integer.

4.

The provided SQL file model_data.`sql` contains 8 insert statements to add sample data into the `Products`, `Suppliers`, and `StockMovements` tables. Execute each insert statement individually, and after running each one, verify that the data has been inserted correctly by querying the relevant table. Capture and submit screenshots of the output after each insertion to demonstrate the successful insertion of data.

5.

Write a SQL statement to create an index on the product_id column in the StockMovements table to improve query performance when filtering or joining on this column.

6.

Write a SQL query to find all suppliers who have supplied products with a quantity greater than 50 in any stock movement.

7.

Write a stored procedure named GetStockMovementsBySupplier that takes a supplier's name as input and returns all stock movements for that supplier, including product name, supplier name, movement type, quantity, and movement date.

8.

Write a SQL query to count the total number of products supplied by each supplier. The result should include supplier name and total product count.

9.

Create an index on the `quantity` column in the `StockMovements` table to improve the performance of queries filtering or sorting by quantity. Write the SQL statement to create this index and briefly explain why this index would be useful.

10.

Write a stored procedure named `GetProductStock` that accepts a product ID as input and returns the current stock level for that product, calculated from all stock movements. Include the SQL code for creating the stored procedure and an example call to test it.