**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.

Use the provided SQL file named model\_data.sql, which contains insert statements to add sample data into the Products, Suppliers, and StockMovements tables. Execute the file to insert all records in one go into the respective tables.

5.Create a simple index on the product\_name column in the Products table to improve search performance by name.

6. Create a composite index on the product\_id and movement\_type columns in the StockMovements table to optimize queries that filter by product and movement type together.

7. Create a unique index on the supplier\_name column in the Suppliers table to ensure no two suppliers have the same name.

8. Drop the index idx\_products\_name from the Products table.

9. Write a SQL query to check all indexes defined on the StockMovements table.

10. Write Down Following stored procedures for each task

|  |  |  |  |
| --- | --- | --- | --- |
| **Procedure Name** | **Input(s)** | **Output(s)** | **Task Description** |
| TotalStockIn | product\_id INT | total\_in INT | Return total stock IN quantity for a product |
| TotalStockOut | product\_id INT | total\_out INT | Return total stock OUT quantity for a product |
| NetStockBalance | product\_id INT | net\_stock INT | Calculate net stock (IN - OUT) for a product |
| InsertStockMovement | product\_id INT, quantity INT, movement\_type VARCHAR(10) | – | Insert a new stock movement record |
| UpdateProductPrice | product\_id INT, new\_price DECIMAL(8,2) | – | Update product price |
| SupplierProductCount | supplier\_id INT | product\_count INT | Count products supplied by a specific supplier |
| ProductsBySupplier | supplier\_name VARCHAR(100) | total INT | Count products using supplier name |
| AdjustSupplierID | INOUT supplier\_id INT | (modified input) | Modify supplier ID if less than 1000 |
| StockMovementsByProduct | product\_id INT | (uses SELECT) | Display all stock movements for a given product |
| DeleteStockByProduct | product\_id INT | – | Delete all stock movements of a product |