Module 8

What to learn

- Indexes
  - Clustered Indexes
  - NonClustered Indexes

Practice Exercise

Practice 1

Do the hands on the explained in video

Practice 2

Practice Exercise: Declare and Use a Cursor

Create a cursor to fetch the details of all employees with a salary greater than 5000 from the following table:

EmployeeID	FirstName	LastName	Salary
1	John	Smith	4500
2	Jane	Doe	6200
3	Emily	Johnson	7200
4	Michael	Williams	4900
5	Sarah	Brown	5600

Steps to perform:

- Declare a cursor to select employees with a salary greater than 5000.
- Open the cursor.
- Fetch the data from the cursor one row at a time and print it.
- Close the cursor.
- Deallocate the cursor.

Practice 3

Practice Exercise: Cursor for Selecting Specific Rows

Using the table **Products** provided below, declare a cursor to fetch only those products whose price is greater than 20 and quantity available is more than 50:

ProductID	ProductName	Price	Quantity
1	Pen	5	100
2	Notebook	25	75
3	Stapler	30	30
4	Pencil	2	150
5	Marker	50	60

Implement the following steps:

Declare the cursor to select the required rows.
Open the cursor.
Fetch rows one by one and print them.
Close and deallocate the cursor.

Practice 4

Practice Exercise: Multiple Cursor Fetch

Using the following **Orders** table, declare a cursor and fetch the order details one by one:

OrderID	CustomerName	OrderAmount
101	John Smith	1200
102	Jane Doe	850
103	Emily Johnson	670
104	Michael Williams	1420

Steps:

Declare the cursor to fetch all columns from the table.
Open the cursor.
Fetch the rows one at a time using a loop and print them in a structured format.
Close and deallocate the cursor.

Practice 5

Practice Exercise: Close and Reopen a Cursor

Using the table **Students** below, implement a cursor that fetches the data of students one by one. After fetching the first two rows, close the cursor and reopen it to fetch the remaining rows:

StudentID	StudentName	Age
1	Adam	20
2	Jessica	23
3	Eve	21
4	Liam	22

Steps:

Declare and open the cursor.
Fetch the first two rows and then close the cursor.
Reopen the cursor and fetch the remaining rows.
Print the fetched rows properly.

Practice 6

Practice Exercise: Dynamic Cursor Query

Using the following **Sales** table, declare a cursor dynamically to fetch sales data for a specific date range provided by the user during runtime:

SaleID	ProductName	SaleDate	SaleAmount
1	Tablet	2023-09-15	500
2	Phone	2023-09-16	1500
3	Laptop	2023-09-18	2500
4	Smartwatch	2023-09-19	800

Steps:

Prompt the user for input for StartDate and EndDate.
Declare a dynamic cursor query to filter rows based on the user input range.
Fetch and print the data row by row.
Close and deallocate the cursor.

Assignment Exercise

Assignment 1

SQL MTA book uploaded on tutorial site.

Do the assignment of the chapter 4 only for Indexes.

Assignment 2

Assignment: Inventory Management System - Using Cursors

Scenario: A small retail store wants to manage its inventory efficiently. The store requires SQL to handle customer orders, update inventory automatically after each transaction, and generate reports dynamically. Use cursors to implement the functionality.

Instructions:

Using the tables below, perform the following tasks:

ProductID	ProductName	Stock	PricePerUnit
1	Pen	150	5
2	Notebook	80	25
3	Eraser	200	2
4	Markers	50	30
5	Stapler	20	150

Inventory

OrderID	CustomerName	ProductID	Quantity	OrderDate
101	John Smith	2	5	2023-09-15
102	Jane Doe	5	2	2023-09-16
103	Emily Johnson	1	10	2023-09-18
104	Michael Williams	4	3	2023-09-19

Orders

Tasks:

Update inventory: Implement a cursor to iterate over the **Orders** table and reduce the stock in the **Inventory** table based on the quantity ordered for each product.
Stock report: Generate a list of all products that are either out of stock or have less than 10 units remaining.
Revenue calculation: Calculate the total revenue generated for each product after processing the orders.
Customer order details: Using a cursor, generate a report that lists each customer's total spending on their respective orders and the order details.
Write the SQL queries and provide explanations of the logic used in each step.
Ensure to open, fetch, close, and deallocate the cursor properly at each step.
Handle any errors, such as stock not being available, using exception handling in SQL.

Online Reference

No online Reference

Supported Files

SQL MTA.pdf

Introduction to Relational Databases

Introduction to Select Statement

Filtering Results with WHERE Statements

Utilizing Joins

Executing Sub queries and Unions

Aggregating Data

- Advanced Data Aggregations
- Built in Functions
- Query Optimization
- Modifying Data
- Advanced Data Modification
- Stored Procedure
- Transaction
- Error handling
- Designing Tables
- triggers