EASTERN INTERNATIONAL UNIVERSITY **Practice Assignment – Quarter 4, 2023-2024**

**SCHOOL OF COMPUTING Course Name:** Database

**AND INFORMATION TECHNOLOGY** **Course Code:** CSE 301

🙙🕮🙛 **Student’s Full Name:**

**Practice Assignment 7**

**Student ID:**

***Instruction*:**

*\* Students are allowing to write their answers (like SQL queries, Screen shot of outputs, etc.) in word file (Answer sheet) provided by instructor. After finishing the assignment, students must convert the word file (Answer sheet) into a PDF file. Finally, students upload the file in Moodle.*

1. Create the following tables in a new database ‘Assignment3’:

Clients(**Client\_Number**, Client\_Name, Address, City, Pincode, Province, Amount\_Paid, Amount\_Due)

Product(**Product\_Number**, Product\_Name, Quantity\_On\_Hand, Quantity\_Sell, Sell\_Price, Cost\_Price)

Salesman (**Salesman\_Number**, Salesman \_Name, Address, City, Pincode, Province, Salary, Sales\_Target, Target\_Achieve, Phone)

Salesorder(**Order\_Number**, Order\_Date, **Client\_Number**, **Salesman\_Number**, Delivery\_Status, Delivery\_Date, Order\_Status)

Salesorderdetails(**Order\_Number, Product\_Number**, Order\_Quantity)

1. *SQL UNION*

**Syntax:**

**SELECT *column\_name(s)* FROM *table1*  
UNION  
SELECT *column\_name(s)* FROM *table2*;**

**The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:**

**SELECT *column\_name(s)* FROM *table1*  
UNION ALL  
SELECT *column\_name(s)* FROM *table2*;**

1. SQL statement returns the cities (only distinct values) from both the "Clients" and the "salesman" table.
2. SQL statement returns the cities (duplicate values also) both the "Clients" and the "salesman" table.
3. SQL statement returns the Ho Chi Minh cities (only distinct values) from the "Clients" and the "salesman" table.
4. SQL statement returns the Ho Chi Minh cities (duplicate values also) from the "Clients" and the "salesman" table.
5. SQL statement lists all Clients and salesman.
6. Write a SQL query to find all salesman and clients located in the city of Ha Noi on a table with information: ID, Name, City and Type.
7. Write a SQL query to find those salesman and clients who have placed more than one order. Return ID, name and order by ID.
8. Retrieve Name, Order Number (order by order number) and Type of client or salesman with the client names who placed orders and the salesman names who processed those orders.
9. Write a SQL query to create a union of two queries that shows the salesman, cities, and target\_Achieved of all salesmen. Those with a target of 60 or greater will have the words 'High Achieved', while the others will have the words 'Low Achieved'.
10. Write query to creates lists all products (Product\_Number AS ID, Product\_Name AS Name, Quantity\_On\_Hand AS Quantity) and their stock status. Products with a positive quantity in stock are labeled as 'More 5 pieces in Stock'. Products with zero quantity are labeled as ‘Less 5 pieces in Stock'.
11. *STORE PROCEDURES*

|  |
| --- |
| ***Statements:***   1. Create a procedure stores   **Delimiter $$**  **CREATE PROCEDURE sp\_name () .../**  **CREATE PROCEDURE sp\_name ([IN] param\_name type).../**  **CREATE PROCEDURE sp\_name ([OUT] param\_name type).../**  **CREATE PROCEDURE sp\_name ([INOUT] param\_name type)...**  **Begin**  **Select statements;**  **End$$**  **Delimiter ;**   1. Call a procedure stores   **Call** *name\_store* (*value of parameter if have*);   1. Drop a procedure stores   **Drop procedure** *name\_store*; |

1. Create a procedure stores get\_clients \_by\_city () saves the all Clients in table. Then Call procedure stores.
2. Drop get\_clients \_by\_city () procedure stores.
3. Create a stored procedure to update the delivery status for a given order number. Change value delivery status of order number “O20006” and “O20008” to “On Way”.
4. Create a stored procedure to retrieve the total quantity for each product.
5. Create a stored procedure to update the remarks for a specific salesman.
6. Create a procedure stores find\_clients() saves all of clients and can call each client by client\_number.
7. Create a procedure stores salary\_salesman() saves all of clients (salesman\_number, salesman\_name, salary) having salary >15000. Then execute the first 2 rows and the first 4 rows from the salesman table.
8. Procedure MySQL MAX() function retrieves maximum salary from MAX\_SALARY of salary table.
9. Create a procedure stores execute finding amount of order\_status by values order status of salesorder table.
10. Create a stored procedure to calculate and update the discount rate for orders.
11. Count the number of salesman with following conditions : SALARY < 20000; SALARY > 20000; SALARY = 20000.
12. Create a stored procedure to retrieve the total sales for a specific salesman.
13. Create a stored procedure to add a new product:

**Input variables:** Product\_Number, Product\_Name, Quantity\_On\_Hand, Quantity\_Sell, Sell\_Price, Cost\_Price.

1. Create a stored procedure for calculating the total order value and classification:

* This stored procedure receives the order code (p\_Order\_Number) và return the total value (p\_TotalValue) and order classification (p\_OrderStatus).
* Using the cursor (CURSOR) to browse all the products in the order (SalesOrderDetails ).
* LOOP/While: Browse each product and calculate the total order value.
* CASE WHEN: Classify orders based on total value:

Greater than or equal to 10000: "Large"

Greater than or equal to 5000: "Midium"

Less than 5000: "Small"