Labsheet -8

Ques 1 → Cursor to iterate through all Customers and Print CNR and Price.

```
DELIMITER $$
CREATE PROCEDURE Display_Customers()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE vCNR INT;
    DECLARE vPrice DECIMAL(10,2);
    DECLARE cur CURSOR FOR SELECT CNR, Price FROM Customer;
    DECLARE CONTIUE HANDLER FOR NOT FOUND SET done=1;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO vCNR, vPrice;
        IF done THEN LEAVE read_loop; END IF;
        SELECT CONCAT('Customer ID:', vCNR, ', Price:',
            vPrice) AS Customer_Info;
    END LOOP;
    CLOSE cur;
END$$
DELIMITER;
```

Ques 2 → Cursor to Find orders where Quantity > 2

```
DELIMITER $$
CREATE PROCEDURE Display_High_Quantity_Order();
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE vCNR, vANR, vQty INT;
    DECLARE cur CURSOR FOR SELECT customer_
        CNR, Article_ANR, Quantity
```

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=1;
OPEN cur;
read_loop: LOOP
    FETCH cur INTO vCNR, vANR, vQty;
    IF done THEN LEAVE read_loop; END IF;
    SELECT CONCAT('Customer:', vCNR, ', Quantity:', vQty)
                AS High_Quantity.
END LOOP;
CLOSE cur;
END $$
DELIMITER ;
```

Ques 3 → Stored Procedure to insert a new article.

```
DELIMITER $$
CREATE PROCEDURE Add_Article (IN p_ANR INT, INp_Name
        VARCHAR(100), INp_Price DECIMAL(10,2))
BEGIN
    INSERT INTO Article (ANR, Name, Price) VALUES(p_AND,
            P_Name, p_Price);
END $$
DELIMITER;
```

Ques 4 → Procedure to get all orders for a given customer

```
DELIMITER $$
CREATE PROCEDURE Get_Orders_By_Customer(Inp_CNR INT)
BEGIN
    SELECT * FROM OPOS WHERE Customer_CNR = p_CNR;
END $$
DELIMITER;
```

Ques 5 → Procedure to increase customer price by 10%.

```
DELIMITER $$
CREATE PROCEDURE Increase_Customer_Price (IN p_CNR
                    INT)
    BEGIN
        UPDATE Customer SET Price = Price * 1.10 WHERE
                    CNR = p_CNR;
    END $$
DELIMITER;
```

Ques 6 → Function to return total quantity of an article

```
DELIMITER $$
CREATE FUNCTION Total_Quantity_Article (p_ANR INT)
RETURNS INT DETERMINISTIC
BEGIN
    DECLARE totalQty INT;
    SELECT Sum(Quantity) INTO totalQty FROM OPOS
            WHERE Article_AND = p_ANR;
    RETURN IFNULL (totalQty, 0);
END $$
DELIMITER;
```

Ques 7 → Function to Calculate total amount Spent by a customer.

```
DELIMITER $$
CREATE FUNCTION Total_Amount_By_Customer
                    (p_CNR INT)
RETURNS DECIMAL(10,2) DETERMINISTIC
```

```
BEGIN
    DECLARE totalAmt DECIMAL(10,2);
    SELECT SUM(o.Quantity * a.Price) INTO totalAmt
    FROM OPOS o INNER JOIN Article a ON o.Article_ANR = a.ANR
    WHERE o.Customer_CNR = p_CNR;
    RETURN IFNULL(totalAmt, 0);
END$$
DELIMITER;
```

Ques8 → Trigger to update lastUpdated when Price changes.

```
ALTER TABLE Customer ADD COLUMN LastUpdated DATETIME;
DELIMITER $$
CREATE TRIGGER Update_Customer_LastUpdated
BEFORE UPDATE ON Customer
FOR EACH ROW
BEGIN
    IF NEW.Price <> OLD.Price THEN
        SET NEW.LastUpdated = NOW();
    END IF;
END$$
DELIMITER;
```

Ques9 → Trigger to prevent inserting order if Quantity < 1.

```
DELIMITER $$
CREATE TRIGGER Prevent_Invalid_Quantity
BEFORE INSERT ON OPOS
FOR EACH ROW
BEGIN
    IF NEW.Quantity < 1 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
            'Quantity must be at least 1';
    END IF;
END$$
DELIMITER;
```

Ques10: Trigger to update Stock table when a new order is inserted?

```
CREATE TABLE Stock (Article_ANR INT PRIMARY KEY,
                    Quantity_Remaining INT);

DELIMITER $$
CREATE TRIGGER Update_Stock_After_Order
 AFTER INSERT ON OPOS
 FOR EACH ROW
 BEGIN
      UPDATE Stock SET Quantity_Remaining = Quantity_Remaining
                                           - NEW.Quantity
      WHERE Article_ANR = NEW.Article_ANR;
 END $$
 DELIMITER;
```

Ques11: Partition OPOS table by range of ODate

```
CREATE TABLE OPOS_Partitioned(
      OPOS_ID INT PRIMARY KEY AUTO_INCREMENT,
      Customer_CNR INT, Article_ANR INT, Quantity INT,
            ONR INT, ODate DATE
)
PARTITION BY RANGE (YEAR(ODate))(
    PARTITION p2024 VALUES LESS THAN (2025),
    PARTITION p2025 VALUES LESS THAN (2026),
    PARTITION pmax VALUES LESS THAN MAXVALUE
);
```

Ques12 → Partition Article table by Price range

```
CREATE TABLE Article_Partitioned (
    ANR INT PRIMARY KEY, Name VARCHAR(100), Price
        DECIMAL(10,2)
)
PARTITION BY RANGE (Price) (
    PARTITION low_price VALUES LESS THAN (101),
    PARTITION mid_price VALUES LESS THAN (501),
    PARTITION high_price VALUES LESS THAN MAXVALUE
);
```