ADVANCED SQL.......

QUESTION-1

SOLUTION-

```
                    WITH ExampleCTE AS (
   SELECT ProductID, ProductName
   FROM Products
)
SELECT * FROM ExampleCTE;
```

QUESTION-2

SOLUTION-2

```
                    - Updatable view example
CREATE VIEW vw_UpdatableProducts AS
SELECT ProductID, ProductName, Price
FROM Products;

-- Read-only view example
CREATE VIEW vw_ReadOnlyProducts AS
SELECT Category, COUNT(*) AS TotalProducts
FROM Products
GROUP BY Category;
```

QUESTION-3

SOLUTION-3

```
               -- Example stored procedure structure
DELIMITER //
CREATE PROCEDURE SampleProcedure()
```

```
BEGIN
    SELECT * FROM Products;
END //
DELIMITER ;
```

QUESTION-4

SOLUTION-4

```
-- Example trigger use case (AFTER DELETE auditing)
DELIMITER //
CREATE TRIGGER trg_AuditProductDelete
AFTER DELETE ON Products
FOR EACH ROW
BEGIN
    INSERT INTO ProductArchive
    (ProductID, ProductName, Category, Price, DeletedAt)
    VALUES
    (OLD.ProductID, OLD.ProductName, OLD.Category, OLD.Price, NOW());
END //
DELIMITER ;
```

QUESTION-5

ANSWER-5

```
-- Example of normalized structure using foreign keys

-- Parent table
CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(50)
);

-- Child table
CREATE TABLE Products_Normalized (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    CategoryID INT,
    Price DECIMAL(10,2),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);
```

QUESTION-6

SOLUTION-6

```
                    WITH RevenueCTE AS (
    SELECT
        p.ProductID,
        p.ProductName,
        SUM(p.Price * s.Quantity) AS Revenue
    FROM Products p
    JOIN Sales s ON p.ProductID = s.ProductID
    GROUP BY p.ProductID, p.ProductName
)
SELECT *
FROM RevenueCTE
WHERE Revenue > 3000;
```

QUESTION-7

ANSWER-7

```
                CREATE VIEW vw_CategorySummary AS
SELECT
   Category,
   COUNT(*) AS TotalProducts,
   AVG(Price) AS AveragePrice
FROM Products
GROUP BY Category;
```

QUESTION-8

SOLUTION-8

```
                  CREATE VIEW vw_ProductPrice AS
SELECT ProductID, ProductName, Price
FROM Products;

UPDATE vw_ProductPrice
SET Price = 1300
WHERE ProductID = 1;
```

QUESTION-9

SOLUTION-9

```
                 DELIMITER //
CREATE PROCEDURE GetProductsByCategory(IN p_Category VARCHAR(50))
BEGIN
```

```
   SELECT *
   FROM Products
   WHERE Category = p_Category;
END //
DELIMITER ;
```

QUESTION-10

ANSWER-10

```
                CREATE TABLE ProductArchive (
   ProductID INT,
   ProductName VARCHAR(100),
   Category VARCHAR(50),
   Price DECIMAL(10,2),
   DeletedAt TIMESTAMP
);

DELIMITER //
CREATE TRIGGER trg_AfterProductDelete
AFTER DELETE ON Products
FOR EACH ROW
BEGIN
   INSERT INTO ProductArchive
   (ProductID, ProductName, Category, Price, DeletedAt)
   VALUES
   (OLD.ProductID, OLD.ProductName, OLD.Category, OLD.Price, NOW());
END //
DELIMITER ;
```