Activity: Writing Basic SQL Queries with Microsoft Copilot

```
SELECT
   ProductName,
   Category,
   Price,
   StockLevel
FROM
   Products;


SELECT
   ProductName,
   Category,
   Price,
   StockLevel
FROM
   Products
WHERE
   Category = 'Electronics'  -- Replace 'Electronics' with the desired category
ORDER BY
   Price ASC;  -- Sorting by Price in ascending order


SELECT
   ProductName,
   Category,
   Price,
   StockLevel
FROM
   Products
WHERE
```

```
    StockLevel <= 10  -- Adjust threshold as needed
ORDER BY
    Price ASC;  -- Sorting by Price in ascending order
```

Activity 2: Creating Complex SQL Queries with Microsoft Copilot

```
SELECT
    p.ProductName,
    s.SaleDate,
    st.StoreLocation,
    s.UnitsSold
FROM
    Sales s
JOIN
    Products p ON s.ProductID = p.ProductID
JOIN
    Stores st ON s.StoreID = st.StoreID
ORDER BY
    s.SaleDate DESC; -- Sorting by most recent sales first


SELECT
    p.ProductName,
    SUM(s.UnitsSold) AS TotalUnitsSold
FROM
    Sales s
JOIN
    Products p ON s.ProductID = p.ProductID
GROUP BY
    p.ProductName
ORDER BY
    TotalUnitsSold DESC; -- Sorting to show top-selling products first
```

```
SELECT
   sp.SupplierName,
   COUNT(*) AS DelayedDeliveries
FROM
   Deliveries d
JOIN
   Suppliers sp ON d.SupplierID = sp.SupplierID
WHERE
   d.DeliveryDate > d.ExpectedDeliveryDate  -- Finding delayed deliveries
GROUP BY
   sp.SupplierName
ORDER BY
   DelayedDeliveries DESC; -- Sorting by the highest delays first
```

Activity 3: Debugging and Optimizing SQL Queries with Microsoft Copilot

```
SELECT
   p.ProductName,
   s.SaleDate,
   st.StoreLocation,
   s.UnitsSold
FROM
   Sales s
LEFT JOIN
   Products p ON s.ProductID = p.ProductID
LEFT JOIN
   Stores st ON s.StoreID = st.StoreID
ORDER BY
   s.SaleDate DESC;
```

```
SELECT
    p.ProductName,
    SUM(s.UnitsSold) AS TotalUnitsSold
FROM
    Products p
JOIN
    Sales s ON p.ProductID = s.ProductID
GROUP BY
    p.ProductName
ORDER BY
    TotalUnitsSold DESC;
-- Index for filtering and joining Sales table
CREATE INDEX idx_sales_productid ON Sales (ProductID);
CREATE INDEX idx_sales_storeid ON Sales (StoreID);
CREATE INDEX idx_sales_saledate ON Sales (SaleDate);

-- Index for filtering by category in Products
CREATE INDEX idx_products_category ON Products (Category);

-- Index for joining Deliveries with Suppliers
CREATE INDEX idx_deliveries_supplierid ON Deliveries (SupplierID);

-- Index for checking delayed deliveries
CREATE INDEX idx_deliveries_dates ON Deliveries (DeliveryDate, ExpectedDeliveryDate);
SELECT
    p.ProductName,
    COALESCE(SUM(s.UnitsSold), 0) AS TotalUnitsSold
FROM
    Products p
```

```
LEFT JOIN
    Sales s ON p.ProductID = s.ProductID
GROUP BY
    p.ProductName
ORDER BY
    TotalUnitsSold DESC;
```