```
                                -- Data Cleaning
-- Fixing NULLs in Non-Nullable Important Fields
-- 1.1 Fix NULL sale_date by setting to today's date
UPDATE sales
SET sale_date = CAST(GETDATE() AS DATE)
WHERE sale_date IS NULL;


-- 1.2 Fix NULL product_id by assigning to a default 'Unknown Product'
IF NOT EXISTS (SELECT * FROM products WHERE product_name = 'Unknown Product')
BEGIN
    INSERT INTO products (product_name, category_id)
    VALUES ('Unknown Product', (SELECT TOP 1 category_id FROM categories));
END

UPDATE sales
SET product_id = (SELECT product_id FROM products WHERE product_name = 'Unknown
  Product')
WHERE product_id IS NULL;


-- 1.3 Fix NULL rep_id by assigning to a default 'Unknown Rep'
IF NOT EXISTS (SELECT * FROM sales_representatives WHERE rep_name = 'Unknown
  Rep')
BEGIN
    INSERT INTO sales_representatives (rep_name, region_id)
    VALUES ('Unknown Rep', (SELECT TOP 1 region_id FROM regions));
END

UPDATE sales
SET rep_id = (SELECT rep_id FROM sales_representatives WHERE rep_name = 'Unknown
   Rep')
WHERE rep_id IS NULL;

GO


-- Fixing Foreign Key Violations by Assigning Default Values
-- -- 2.1 Fix sales records with invalid customer_id by assigning a dummy
  customer 'Unknown'
IF NOT EXISTS (SELECT * FROM customers WHERE customer_type = 'Regular')
BEGIN
    INSERT INTO customers (customer_type)
    VALUES ('Regular');
END

UPDATE sales
SET customer_id = (SELECT TOP 1 customer_id FROM customers WHERE customer_type =
   'Regular')
WHERE customer_id IS NOT NULL
AND customer_id NOT IN (SELECT customer_id FROM customers);
```

```sql
-- 2.2 Fix invalid payment_method_id
IF NOT EXISTS (SELECT * FROM payment_methods WHERE method_name = 'Cash')
BEGIN
    INSERT INTO payment_methods (method_name)
    VALUES ('Cash');
END

UPDATE sales
SET payment_method_id = (SELECT TOP 1 payment_method_id FROM payment_methods
  WHERE method_name = 'Cash')
WHERE payment_method_id IS NOT NULL
AND payment_method_id NOT IN (SELECT payment_method_id FROM payment_methods);

GO


-- Correcting Negative or Invalid Quantity, Price, and Sales Values
-- 3.1 Correcting negative quantities to 0
UPDATE sales
SET quantity_sold = 0
WHERE quantity_sold < 0;

-- 3.2 Correcting zero or negative unit prices
UPDATE sales
SET unit_price = 10.00
WHERE unit_price <= 0;

-- 3.3 Correcting zero or negative total_sales by recalculating from quantity
  and unit price
UPDATE sales
SET total_sales = quantity_sold * unit_price
WHERE total_sales <= 0;

-- 3.4 Forcing recalculation of total_sales where mismatch > 0.01
UPDATE sales
SET total_sales = quantity_sold * unit_price
WHERE ABS((quantity_sold * unit_price) - total_sales) > 0.01;

GO


-- Fixing Category or Product Assignment Issues
-- 4.1 Assigning products without valid category to a 'Miscellaneous' category
IF NOT EXISTS (SELECT * FROM categories WHERE category_name = 'Miscellaneous')
BEGIN
    INSERT INTO categories (category_name)
    VALUES ('Miscellaneous');
END

UPDATE products
SET category_id = (SELECT category_id FROM categories WHERE category_name =
  'Miscellaneous')
WHERE category_id NOT IN (SELECT category_id FROM categories);
```

```
GO


-- Fixing Sales Rep Region Assignment Issues
-- 5.1 Assigning reps without valid regions to any available region
UPDATE sales_representatives
SET region_id = (SELECT TOP 1 region_id FROM regions)
WHERE region_id NOT IN (SELECT region_id FROM regions);

GO


-- Removing Exact Duplicate Sales Rows
-- 7.1 Optional: Deleting exact duplicate sales if detected
WITH DuplicateSales AS (
    SELECT
        MIN(sale_id) AS KeepSaleID,
        sale_date,
        product_id,
        rep_id,
        customer_id,
        COUNT(*) AS DuplicateCount
    FROM sales
    GROUP BY sale_date, product_id, rep_id, customer_id
    HAVING COUNT(*) > 1
)
DELETE FROM sales
WHERE sale_id NOT IN (SELECT KeepSaleID FROM DuplicateSales)
AND EXISTS (
    SELECT 1
    FROM DuplicateSales d
    WHERE
        d.sale_date = sales.sale_date
        AND d.product_id = sales.product_id
        AND d.rep_id = sales.rep_id
        AND d.customer_id = sales.customer_id
);

GO
```