```sql
-- SQL Server MERGE: MERGE statement used to update data in a table based on values matched from another table.
/*
Syntax:
MERGE target_table USING source_table
ON merge_condition
WHEN MATCHED
    THEN update_statement
WHEN NOT MATCHED
    THEN insert_statement
WHEN NOT MATCHED BY SOURCE
    THEN DELETE;
*/

-- Creating New Tables
CREATE TABLE sales.category (
    category_id INT PRIMARY KEY,
    category_name VARCHAR(255) NOT NULL,
    amount DECIMAL(10 , 2 )
);

INSERT INTO sales.category(category_id, category_name, amount)
VALUES(1,'Children Bicycles',15000),
    (2,'Comfort Bicycles',25000),
    (3,'Cruisers Bicycles',13000),
    (4,'Cyclocross Bicycles',10000);


CREATE TABLE sales.category_staging (
    category_id INT PRIMARY KEY,
    category_name VARCHAR(255) NOT NULL,
    amount DECIMAL(10 , 2 )
);
```

```sql
INSERT INTO sales.category_staging(category_id, category_name, amount)
VALUES(1,'Children Bicycles',15000),
    (3,'Cruisers Bicycles',13000),
    (4,'Cyclocross Bicycles',20000),
    (5,'Electric Bikes',10000),
    (6,'Mountain Bikes',10000);


SELECT * FROM sales.category;
SELECT * FROM sales.category_staging;
-- To update data to the sales.category (target table) with the values from the sales.category_staging
--(source table), you use the following MERGE statement:
MERGE sales.category t
USING sales.category_staging s
ON (s.category_id = t.category_id)
WHEN MATCHED
    THEN UPDATE SET
        t.category_name = s.category_name,
        t.amount = s.amount
WHEN NOT MATCHED BY TARGET
    THEN INSERT (category_id, category_name, amount)
    VALUES (s.category_id, s.category_name, s.amount)
WHEN NOT MATCHED BY SOURCE
    THEN DELETE;

------------------------------------------------------------
-- SQL Server Transaction: A transaction is a single unit of work
--that typically contains multiple T-SQL statements.

-- If a transaction is successful, the changes are committed to the database.
--However, if a transaction has an error, the changes have to be rolled back.
CREATE TABLE invoices (
  id int IDENTITY PRIMARY KEY,
  customer_id int NOT NULL,
  total decimal(10, 2) NOT NULL DEFAULT 0 CHECK (total >= 0)
);
```

```sql
CREATE TABLE invoice_items (
  id int,
  invoice_id int NOT NULL,
  item_name varchar(100) NOT NULL,
  amount decimal(10, 2) NOT NULL CHECK (amount >= 0),
  tax decimal(4, 2) NOT NULL CHECK (tax >= 0),
  PRIMARY KEY (id, invoice_id),
  FOREIGN KEY (invoice_id) REFERENCES invoices (id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);

-- Transaction
BEGIN TRANSACTION;

INSERT INTO invoices (customer_id, total)
VALUES (100, 0);

INSERT INTO invoice_items (id, invoice_id, item_name, amount, tax)
VALUES (10, 1, 'Keyboard', 70, 0.08),
       (20, 1, 'Mouse', 50, 0.08);

UPDATE invoices
SET total = (SELECT
  SUM(amount * (1 + tax))
FROM invoice_items
WHERE invoice_id = 1);

COMMIT;
```