10 statements for insertion:

```
INSERT INTO Customers (name, email, phone) VALUES ('John Doe',
'john.doe@email.com', '123456789');
INSERT INTO Customers (name, email, phone) VALUES ('Jane Smith',
'jane.smith@email.com', '987654321');
INSERT INTO Customers (name, email, phone) VALUES ('Tom Hanks',
'tom.hanks@email.com', '123123123');
INSERT INTO Brands (brand_name) VALUES ('Apple');
INSERT INTO Brands (brand_name) VALUES ('Samsung');
INSERT INTO Categories (categories_name) VALUES ('Smartphones');
INSERT INTO Categories (categories_name) VALUES ('Tablets');
INSERT INTO Suppliers (supplier_name, contact_email) VALUES ('Apple
Supplier', 'contact@apple.com');
INSERT INTO Suppliers (supplier_name, contact_email) VALUES ('Samsung
Supplier', 'contact@samsung.com');
INSERT INTO Orders (customer_id, order_date, total_amount, status) VALUES
(1, '2024-01-01', 599.99, 'Pending');
INSERT INTO Orders (customer_id, order_date, total_amount, status) VALUES
(2, '2024-01-05', 899.99, 'Completed');
INSERT INTO Orders (customer_id, order_date, total_amount, status) VALUES
(3, '2024-01-10', 1299.99, 'Shipped');
INSERT INTO Phones (brand_id, category_id, supplier_id, name, price,
release_date) VALUES (1, 1, 1, 'iPhone 15', 999.99, '2023-09-20');
INSERT INTO Phones (brand_id, category_id, supplier_id, name, price,
release_date) VALUES (2, 2, 2, 'Samsung Galaxy S23', 799.99,
'2023-08-15');
```

10 statements for updating:

```
UPDATE Customers SET email = 'john.updated@email.com' WHERE customer_id =
1;
UPDATE Suppliers SET supplier_name = 'Xiaomi Supplier' WHERE suppliers_id
= 1;
UPDATE Phones SET price = 1099.99 WHERE phone_id = 1;
UPDATE OrderItems SET quantity = 2 WHERE order_id = 1 AND phone_id = 1;
UPDATE Orders SET status = 'Shipped' WHERE order_id = 1;
UPDATE Brands SET brand_name = 'Updated Apple' WHERE brand_id = 1;
UPDATE Phones SET release_date = '2023-10-01' WHERE phone_id = 1;
UPDATE Phones SET category_id = 2 WHERE phone_id = 2;
UPDATE Orders SET total_amount = 1199.99 WHERE order_id = 1;
UPDATE Customers SET phone = '111222333' WHERE customer_id = 2;
```

10 statements for deletions:

```
DELETE FROM Orders WHERE customer_id = 1;
DELETE FROM Customers WHERE customer_id = 1;
DELETE FROM Phones WHERE phone_id = 2;
DELETE FROM Orders WHERE order_id = 2;
DELETE FROM OrderItems WHERE order_id = 1 AND phone_id = 1;
DELETE FROM Phones WHERE brand_id = 2;
DELETE FROM Brands WHERE brand_id = 2;
DELETE FROM Phones WHERE category_id = 1;
DELETE FROM Categories WHERE categories_id = 1;
DELETE FROM Orders WHERE total_amount > 1000;
DELETE FROM Phones WHERE release_date < '2023-01-01';
```

5 alter table:

```
ALTER TABLE Customers CHANGE COLUMN name full_name VARCHAR(100);
ALTER TABLE Phones MODIFY price DECIMAL(12, 2);
ALTER TABLE Orders ADD COLUMN created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP;
ALTER TABLE Suppliers ADD COLUMN last_updated TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;
ALTER TABLE Suppliers MODIFY contact_email VARCHAR(150);
```

1 big statement to join all tables in the database:

```
SELECT
    Customers.full_name AS customer_name,
    Orders.order_id,
    Orders.order_date,
    Phones.name AS phone_name,
    OrderItems.quantity,
    OrderItems.price,
    Brands.brand_name,
    Categories.categories_name,
    Suppliers.supplier_name
FROM
    Orders
JOIN
    OrderItems ON Orders.order_id = OrderItems.order_id
JOIN
    Phones ON OrderItems.phone_id = Phones.phone_id
JOIN
    Brands ON Phones.brand_id = Brands.brand_id
JOIN
    Categories ON Phones.category_id = Categories.categories_id
JOIN
```

```
    Suppliers ON Phones.supplier_id = Suppliers.suppliers_id
JOIN
    Customers ON Orders.customer_id = Customers.customer_id;
```

5 statements with left, right, inner, outer joins:

```
SELECT Customers.full_name, Orders.order_date
FROM Customers
LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;

SELECT Customers.full_name, Orders.order_date
FROM Orders
RIGHT JOIN Customers ON Orders.customer_id = Customers.customer_id;

SELECT Customers.full_name, Orders.order_date
FROM Customers
INNER JOIN Orders ON Customers.customer_id = Orders.customer_id;

SELECT Customers.full_name, Orders.order_date
FROM Customers
LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id
UNION
SELECT Customers.full_name, Orders.order_date
FROM Orders
RIGHT JOIN Customers ON Orders.customer_id = Customers.customer_id;

SELECT Customers.full_name, Phones.name
FROM Customers
CROSS JOIN Phones;
```

7 statements with aggregate functions and group by and without having:

```
SELECT customer_id, COUNT(order_id) AS total_orders FROM Orders GROUP BY
customer_id;

SELECT AVG(price) AS avg_price FROM Phones;

SELECT SUM(quantity) AS total_quantity FROM OrderItems;

SELECT MAX(price) AS max_price FROM Phones;

SELECT MIN(price) AS min_price FROM Phones;

SELECT category_id, COUNT(phone_id) AS total_phones FROM Phones GROUP BY
category_id;

SELECT SUM(total_amount) AS total_sales FROM Orders;
```

7 statements with aggregate functions and group by and with having:

```
SELECT customer_id, COUNT(order_id) AS total_orders FROM Orders GROUP BY
customer_id HAVING total_orders > 1;

SELECT category_id, AVG(price) AS avg_price FROM Phones GROUP BY
category_id HAVING avg_price > 800;

SELECT id_supplier, SUM(quantity) AS total_quantity FROM Shipments GROUP
BY id_supplier HAVING total_quantity > 100;

SELECT brand_id, MAX(price) AS max_price FROM Phones GROUP BY brand_id
HAVING max_price > 900;

SELECT customer_id, SUM(total_amount) AS total_spent FROM Orders GROUP BY
customer_id HAVING total_spent > 2000;

SELECT order_id, SUM(quantity) AS total_quantity FROM OrderItems GROUP BY
order_id HAVING total_quantity > 5;

SELECT category_id, COUNT(phone_id) AS total_phones FROM Phones GROUP BY
category_id HAVING total_phones >= 1;
```