**CREATING TABLES**

CREATE TABLE customers (

customer\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

address VARCHAR(200),

phone\_number VARCHAR(20)

);

CREATE TABLE products (

product\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL

);

CREATE TABLE orders (

order\_id SERIAL PRIMARY KEY,

customer\_id INTEGER REFERENCES customers(customer\_id),

order\_date DATE NOT NULL,

total\_amount DECIMAL(10, 2) NOT NULL

);

CREATE TABLE order\_items (

order\_item\_id SERIAL PRIMARY KEY,

order\_id INTEGER REFERENCES orders(order\_id),

product\_id INTEGER REFERENCES products(product\_id),

quantity INTEGER NOT NULL

);

CREATE TABLE payments (

payment\_id SERIAL PRIMARY KEY,

order\_id INTEGER REFERENCES orders(order\_id),

payment\_date DATE NOT NULL,

amount DECIMAL(10, 2) NOT NULL

);

**INSERTING DATA**

INSERT INTO customers (name, email, address, phone\_number)

VALUES

('Jennifer Adams', 'jenniferadams@example.com', '444 Pine St, City', '555-7890'),

('Robert Wilson', 'robertwilson@example.com', '555 Maple Ave, City', '555-5678'),

('Jessica Brown', 'jessicabrown@example.com', '777 Elmwood Pl, City', '555-1234'),

('Daniel Davis', 'daviddaniel@example.com', '888 Oak St, City', '555-9876'),

('Sophia Thompson', 'sophiathompson@example.com', '123 Cedar Ln, City', '555-4321'),

('Andrew Miller', 'andrewmiller@example.com', '999 Spruce Dr, City', '555-2468'),

('Emma Martinez', 'emmamartinez@example.com', '111 Birch Rd, City', '555-1357'),

('Matthew Clark', 'matthewclark@example.com', '222 Pinecrest Ave, City', '555-8642'),

('Ava Taylor', 'avataylor@example.com', '555 Elmwood Pl, City', '555-5793'),

('William Turner', 'williamturner@example.com', '777 Oakwood Rd, City', '555-7582'),

('Sophie Anderson', 'sophieanderson@example.com', '555 Walnut St, City', '555-1111'),

('Henry Davis', 'henrydavis@example.com', '777 Maplewood Ave, City', '555-2222'),

('Grace Johnson', 'gracejohnson@example.com', '123 Oak Ln, City', '555-3333'),

('Samuel Thompson', 'samuelthompson@example.com', '888 Elmwood Dr, City', '555-4444'),

('Elizabeth Wilson', 'elizabethwilson@example.com', '555 Pinecrest Rd, City', '555-5555'),

('Joseph Miller', 'josephmiller@example.com', '999 Birch Ave, City', '555-6666'),

('Chloe Brown', 'chloebrown@example.com', '111 Cedar St, City', '555-7777'),

('Benjamin Clark', 'benjaminclark@example.com', '222 Elm Pl, City', '555-8888'),

('Mia Martinez', 'miamartinez@example.com', '555 Oakwood Ln, City', '555-9999'),

('David Turner', 'davidturner@example.com', '777 Pine Rd, City', '555-0000');

select\* from customers;

INSERT INTO products (product\_id, name, description, price)

VALUES

(1, 'car', 'Very big', 100000),

(2, 'phone', 'Latest model', 800),

(3, 'laptop', 'High-performance', 1500),

(4, 'watch', 'Waterproof', 200),

(5, 'headphones', 'Wireless', 100),

(6, 'camera', 'DSLR', 1200),

(7, 'television', 'Smart TV', 1500),

(8, 'tablet', 'Portable', 600),

(9, 'gaming console', '4K gaming', 500),

(10, 'smart speaker', 'Voice assistant', 80),

(11, 'monitor', 'High resolution', 300),

(12, 'fitness tracker', 'Heart rate monitor', 50),

(13, 'drone', 'Aerial photography', 1000),

(14, 'smartwatch', 'Fitness tracking', 200),

(15, 'printer', 'Wireless printing', 150),

(16, 'blender', 'Powerful motor', 80),

(17, 'vacuum cleaner', 'Robotic', 300),

(18, 'microwave', 'Compact', 100),

(19, 'toaster', 'Multiple slots', 40),

(20, 'hair dryer', 'Ionic technology', 60);

select \* from products;

INSERT INTO orders (order\_id, customer\_id, order\_date, total\_amount)

VALUES

(1, 1, '2023-07-01', 100.00),

(2, 2, '2023-07-02', 75.50),

(3, 3, '2023-07-03', 200.00),

(4, 4, '2023-07-04', 50.25),

(5, 5, '2023-07-05', 300.00),

(6, 6, '2023-07-06', 150.75),

(7, 7, '2023-07-07', 80.50),

(8, 8, '2023-07-08', 120.00),

(9, 9, '2023-07-09', 250.50),

(10, 10, '2023-07-10', 175.25),

(11, 1, '2023-07-11', 80.00),

(12, 2, '2023-07-12', 60.50),

(13, 3, '2023-07-13', 180.00),

(14, 4, '2023-07-14', 40.25),

(15, 5, '2023-07-15', 270.00),

(16, 6, '2023-07-16', 140.75),

(17, 7, '2023-07-17', 75.50),

(18, 8, '2023-07-18', 100.00),

(19, 9, '2023-07-19', 220.50),

(20, 10, '2023-07-20', 155.25);

select \* from orders;

INSERT INTO payments (order\_id, payment\_date, amount)

VALUES

(1, '2008-11-11', 5400),

(2, '2008-12-01', 3200),

(3, '2009-01-15', 6700),

(4, '2009-02-28', 4800),

(5, '2009-03-05', 7800),

(6, '2009-04-12', 2200),

(7, '2009-05-22', 6100),

(8, '2009-06-03', 4500),

(9, '2009-07-10', 8900),

(10, '2009-08-14', 5100),

(11, '2009-09-25', 3700),

(12, '2009-10-17', 7200),

(13, '2009-11-02', 5900),

(14, '2009-12-06', 4200),

(15, '2010-01-20', 6900),

(16, '2010-02-15', 5300),

(17, '2010-03-27', 7900),

(18, '2010-04-07', 4800),

(19, '2010-05-12', 6600),

(20, '2010-06-18', 5800);

select \* from payments;

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (55, 1, 2, 100);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (56, 1, 3, 50);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (57, 1, 4, 75);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (58, 2, 1, 200);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (59, 2, 3, 150);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (60, 3, 4, 100);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (61, 3, 2, 50);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (62, 4, 1, 75);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (63, 4, 3, 125);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (64, 5, 2, 150);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (65, 5, 4, 100);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (66, 6, 1, 200);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (67, 6, 2, 75);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (68, 7, 3, 100);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (69, 7, 4, 50);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (70, 8, 1, 125);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (71, 8, 2, 150);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (72, 9, 3, 100);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (73, 9, 4, 75);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity) VALUES (74, 10, 1, 200);

select \* from order\_items;

------------------------------------------------------------------------------------------------------------------------------------------

**JOINS**

**Q1)Retrieve all the orders with the customer details**

select orders.order\_id, customers.name, orders.order\_date, orders.total\_amount from customers INNER JOIN orders on

customers.customer\_id=orders.customer\_id;

**Q2)Retrieve all the customers along with their orders if any**

select \* from customers LEFT JOIN orders on

customers.customer\_id=orders.customer\_id;

**Q3) Retrieve all orders along with customers details if available**

select \* from customers RIGHT JOIN orders on

customers.customer\_id=orders.customer\_id;

**Q4)Retrieve all customers and their associative orders including unmatched records**

select \* from customers FULL OUTER JOIN orders on

customers.customer\_id=orders.customer\_id;

select \* from customers LEFT JOIN orders on

customers.customer\_id=orders.customer\_id;

------------------------------------------------------------------------------------------------------------------------------------------

**FUNCTIONS**

**Q5) Calculate the total price of an order based on the quantity and product price:**

CREATE OR REPLACE FUNCTION get\_order\_total(p\_order\_id INTEGER)

RETURNS DECIMAL(10, 2)

AS $$

DECLARE

v\_total DECIMAL(10, 2);

BEGIN

SELECT SUM(order\_items.quantity \* products.price)

INTO v\_total

FROM order\_items

INNER JOIN products ON order\_items.product\_id = products.product\_id

WHERE order\_items.order\_id = p\_order\_id;

RETURN v\_total;

END;

$$ LANGUAGE plpgsql;

SELECT get\_order\_total(1);

**Q6)Get Count of orders for a specific customer**

CREATE OR REPLACE FUNCTION get\_order\_count(c\_customer\_id INTEGER)

RETURNS INTEGER

AS $$

DECLARE

T\_count Integer;

BEGIN

SELECT COUNT(customer\_id) INTO t\_count

FROM orders

WHERE customer\_id = c\_customer\_id;

RETURN T\_count;

END;

$$ LANGUAGE plpgsql;

SELECT get\_order\_count(1);

**Q.7) Calculate the average order amount for a specific customer:**

CREATE OR REPLACE FUNCTION get\_avg\_amount(c\_customer\_id INTEGER)

RETURNS DECIMAL(10,2)

AS $$

DECLARE

Avg\_amt DECIMAL(10,2);

BEGIN

SELECT AVG(total\_amount) INTO Avg\_amt

FROM orders

WHERE customer\_id = c\_customer\_id;

RETURN Avg\_amt;

END;

$$ LANGUAGE plpgsql;

SELECT get\_avg\_amount(1);

**Q8)GENERATE A REPORT SHOWING THE CUSTOMER NAMES AND THEIR CURROSPONDING ORDER COUNTS**

CREATE OR REPLACE FUNCTION generate\_customer\_order\_report()

RETURNS TABLE (customer\_name VARCHAR, order\_count BIGINT)

AS $$

BEGIN

return query

SELECT customers.name, COUNT(orders.order\_id)

FROM customers

LEFT JOIN orders ON customers.customer\_id = orders.customer\_id

GROUP BY customers.customer\_id, customers.name;

END;

$$ LANGUAGE plpgsql;

-- because we want all the columns from the table

SELECT \* FROM generate\_customer\_order\_report();

------------------------------------------------------------------------------------------------------------------------------------------**TRIGGERS**

**Q9)create a trigger to update order date whenever a new order is inserted.**

CREATE OR REPLACE FUNCTION updates\_order\_date()  
RETURNS TRIGGER  
AS $$  
BEGIN  
 NEW.order\_date := CURRENT\_DATE;  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;CREATE TRIGGER updates\_order\_date\_trigger  
BEFORE INSERT ON orders  
FOR EACH ROW  
EXECUTE FUNCTION update\_order\_date();

**Q10) Create a trigger to calculate and update the total amount whenever an order item is inserted, updated, or deleted:**

CREATE OR REPLACE FUNCTION update\_order\_total()

RETURNS TRIGGER

AS $$

BEGIN

UPDATE orders

SET total\_amount = (

SELECT SUM(order\_items.quantity \* products.price)

FROM order\_items

INNER JOIN products ON order\_items.product\_id = products.product\_id

WHERE order\_items.order\_id = NEW.order\_id

)

WHERE orders.order\_id = NEW.order\_id;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update\_order\_total\_trigger

AFTER INSERT OR UPDATE OR DELETE ON order\_items

FOR EACH ROW

EXECUTE FUNCTION update\_order\_total();

**Q10) Create a trigger to log the payment details whenever new payment is inserted.**

CREATE TABLE payments\_logs (

log\_id SERIAL PRIMARY KEY,

payment\_id INTEGER REFERENCES payments(payment\_id),

log\_timestamp TIMESTAMP DEFAULT current\_timestamp,

log\_message TEXT

);

CREATE OR REPLACE FUNCTION log\_payments\_details()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO payments\_logs (payment\_id, log\_message)

VALUES (NEW.payment\_id, 'Payment received. Amount: ' || NEW.amount || ', Date: ' || NEW.payment\_date);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER log\_payments\_details\_trigger

AFTER INSERT ON payments

FOR EACH ROW

EXECUTE FUNCTION log\_payments\_details();

insert into payments values (89,5,'2023-2-12',3456);

select \* from payments\_logs;

CREATE OR REPLACE FUNCTION log\_payment\_details()

RETURNS TRIGGER

AS $$

BEGIN

INSERT INTO payment\_logs (payment\_id, order\_id, payment\_date, amount)

VALUES (NEW.payment\_id, NEW.order\_id, NEW.payment\_date, NEW.amount);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER log\_payment\_details\_trigger

AFTER INSERT ON payments

FOR EACH ROW

EXECUTE FUNCTION log\_payment\_details();

INSERT INTO payments (payment\_id, order\_id, payment\_date, amount)

VALUES (21, 1, '2023-07-09', 100.00);

SELECT \* FROM payment\_logs;

------------------------------------------------------------------------------------------------------------------------------------------**INDEXES**

1. Create an index on the "email" column of the customers table:```sql  
CREATE INDEX idx\_customers\_email ON customers (email);  
**```2. Create a unique index on the "product\_id" column of the products table:```sql**  
CREATE UNIQUE INDEX idx\_products\_product\_id ON products (product\_id);

**```3. Create a composite index on the "customer\_id" and "order\_date" columns of the orders table:```**  
CREATE INDEX idx\_orders\_customer\_id\_order\_date ON orders (customer\_id, order\_date);  
```4. Create a partial index on the "quantity" column of the order\_items table for items with a quantity greater than 10:```

CREATE INDEX idx\_order\_items\_quantity\_gt\_10 ON order\_items (quantity) WHERE quantity > 10;

EXPLAIN ANALYZE SELECT \* FROM order\_items WHERE quantity > 10;

EXPLAIN ANALYZE SELECT \* FROM orders WHERE customer\_id = 1 AND order\_date = '2023-07-09';

