SQL BASIC AND ADVANCED

SQL

FUNCTIONS | STORED PROCEDURES | TRIGGERS |

Table of Content

Download

FUNCTIONS_SP_TRIGGERS.SQL

Sequence of Learning

Functions

To Calculate the Discount

Stored Procedures

INSERT Records - With Total Price calculations

Triggers

Audit Table - log_orderqty_update - trg_log_orderqty_update

Sequence of Execution

Update a quantity will trigger logging

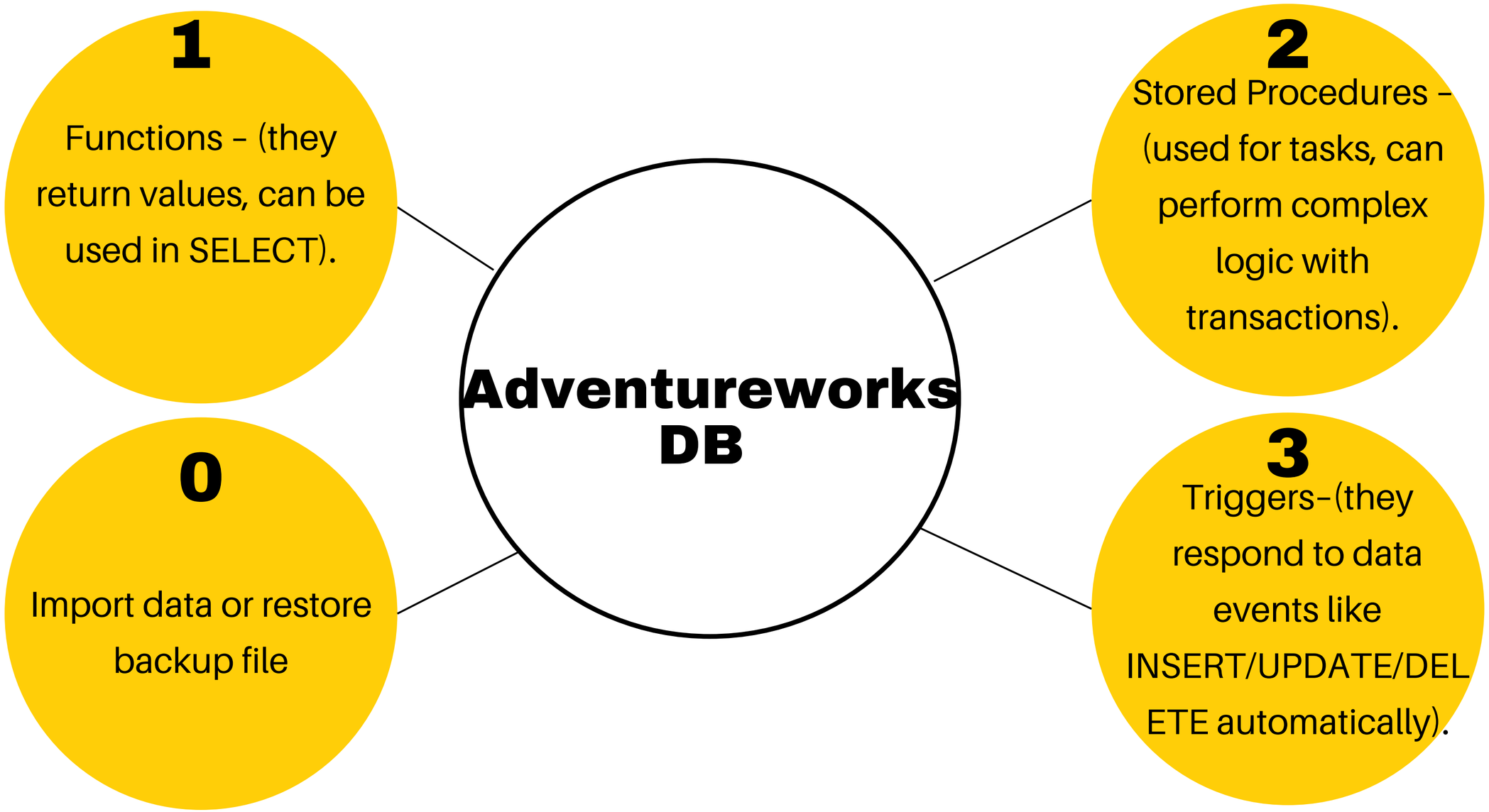
```
UPDATE
sales.salesorderdetail
SET orderqty = 60
WHERE salesorderid =
43659 AND productid =
776;
```

Check the Audit Table

```
SELECT * FROM
audit_orderqty_changes;
```

Output

audit_orderqty_changes


1
Functions – (they return values, can be used in SELECT).
2
Stored Procedures – (used for tasks, can perform complex logic with transactions).
Adventureworks DB
0
Import data or restore backup file
3
Triggers–(they respond to data events like INSERT/UPDATE/DELETE automatically).

Our Business Scenario

Business Goals

DYNAMIC DISCOUNT

Calculate dynamic discounts based on order quantity.

LOG CHANGES

Log changes to order quantity for auditing.

AUTO UPDATE AUDIT LOGS

Automatically update audit logs when orders are updated.

FUNCTIONS

```
CREATE OR REPLACE FUNCTION calculate_discount(order_qty INT)
RETURNS NUMERIC AS $$
BEGIN
  IF order_qty >= 50 THEN
    RETURN 0.15;  -- 15% discount
  ELSIF order_qty >= 20 THEN
    RETURN 0.10;
  ELSE
    RETURN 0.05;
  END IF;
END;
$$ LANGUAGE plpgsql;
```

SALES ORDER DETAIL TABLE

```
SELECT productid, orderqty, calculate_discount(orderqty)
AS discount
FROM sales.salesorderdetail
LIMIT 10;
```

POSTGRESQL STORED PROCEDURES

Stored procedures are used for tasks, such as inserting data or handling logic over multiple steps. They do not return a value directly (like functions) but perform actions.

```
CREATE OR REPLACE PROCEDURE insert_discounted_order(
    p_orderid INT, p_productid INT, p_qty INT, p_unitprice
NUMERIC
)
LANGUAGE plpgsql
AS $$
DECLARE
  v_discount NUMERIC;
  v_total NUMERIC;
BEGIN
  v_discount := calculate_discount(p_qty);
  v_total := p_unitprice * p_qty * (1 - v_discount);

  INSERT INTO sales.salesorderdetail (
    salesorderid, productid, orderqty, unitprice, lineTotal
  ) VALUES (
    p_orderid, p_productid, p_qty, p_unitprice, v_total
  );
END;
$$;
```

CALLING STORED PROCEDURE

```
CALL insert_discounted_order(43659, 776, 25, 100.00,1);
```

TRIGGER - AUDIT TABLE

```
CREATE TABLE audit_orderqty_changes (
   audit_id SERIAL PRIMARY KEY,
   salesorderid INT,
   old_qty INT,
   new_qty INT,
   modified_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP
);
```

POSTGRESQL TRIGGERS

SCENARIO: LOG AUDIT TRAIL WHENEVER ORDERQTY IS UPDATED

TRIGGERS ARE USED TO AUTOMATE ACTIONS WHEN DATA CHANGES (E.G., ON INSERT OR UPDATE). YOU ATTACH THEM TO A TABLE.

TRIGGERS

Scenario: **Log** audit trail whenever **orderqty** is updated

```
CREATE TABLE audit_orderqty_changes (
  audit_id SERIAL PRIMARY KEY,
  salesorderid INT,
  old_qty INT,
  new_qty INT,
        modified_at   TIMESTAMP   DEFAULT
CURRENT_TIMESTAMP
);
```

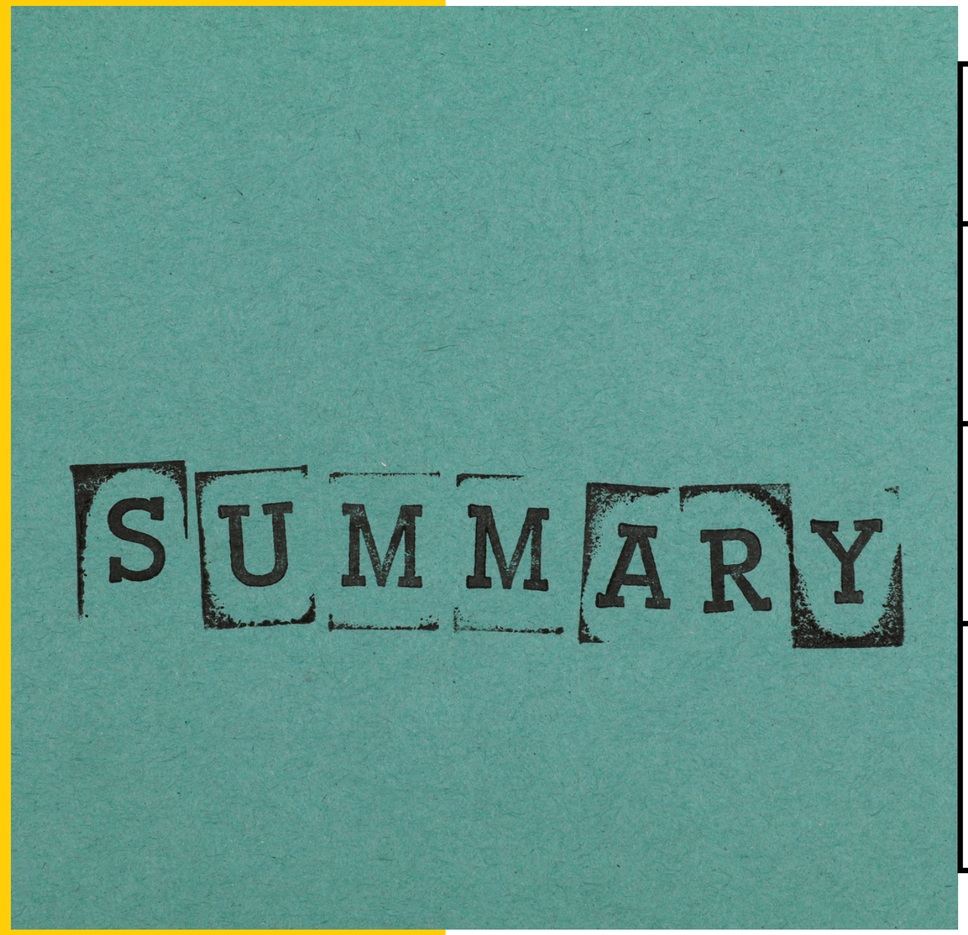
PostgreSQL TRIGGERS

Scenario: Log audit trail whenever orderqty is updated

Create audit table

Create a trigger function

Attach the trigger to the table



Concept	What it Does	When to Use	Returns
Function	Computes and returns a value	Logic in SELECT or reuse	Yes
Stored Procedure	Performs a task or process	Insert/update/ delete logic with conditions	No
Trigger	Responds automatically to table changes	Audits, enforcing rules	No (but executes a function)

Q & A - Discussion