Query | Query History

```
1  ---\Declarative Constraints – Safe Prescriptions
2
3  -- Create schema
4  CREATE SCHEMA healthnet;
5  SET search_path TO healthnet;
6
7  -- Patient table
8  CREATE TABLE patient (
9      id SERIAL PRIMARY KEY,
10     name VARCHAR(100) NOT NULL
11 );
12
```

Data Output | Messages | Notifications

```
SET

Query returned successfully in 400 msec.
```

```
7  -- Patient table
8  CREATE TABLE patient (
9      id SERIAL PRIMARY KEY,
10     name VARCHAR(100) NOT NULL
11 );
12
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 128 msec.
```

```
13 -- Prescription table with constraints
14 CREATE TABLE patient_med (
15     patient_med_id SERIAL PRIMARY KEY,
16     patient_id INT NOT NULL REFERENCES patient(id),
17     med_name VARCHAR(80) NOT NULL,
18     dose_mg NUMERIC(6,2) CHECK (dose_mg >= 0),
19     start_dt DATE,
20     end_dt DATE,
21     CHECK (start_dt IS NULL OR end_dt IS NULL OR start_dt <= end_dt)
22 );
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 142 msec.
```

```
24 -- Passing inserts
25 INSERT INTO patient (name) VALUES ('John Doe');
26 INSERT INTO patient_med (patient_id, med_name, dose_mg, start_dt, end_dt)
27 VALUES (1, 'Amoxicillin', 250.00, '2025-10-01', '2025-10-10');
28
29 -- Failing inserts
30 -- Negative dose
31 INSERT INTO patient_med (patient_id, med_name, dose_mg)
32 VALUES (1, 'Ibuprofen', -100.00);
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 149 msec.
```

```sql
-- Failing inserts
-- Negative dose
INSERT INTO patient_med (patient_id, med_name, dose_mg)
VALUES (1, 'Ibuprofen', -100.00);

-- Inverted dates
INSERT INTO patient_med (patient_id, med_name, dose_mg, start_dt, end_dt)
VALUES (1, 'Ciprofloxacin', 200.00, '2025-11-10', '2025-10-01');
```

Data Output | Messages | Notifications

```
ERROR:  new row for relation "patient_med" violates check constraint "patient_med_dose_mg_check"
Failing row contains (2, 1, Ibuprofen, -100.00, null, null).

SQL state: 23514
Detail: Failing row contains (2, 1, Ibuprofen, -100.00, null, null).
```

```sql
-- Inverted dates
INSERT INTO patient_med (patient_id, med_name, dose_mg, start_dt, end_dt)
VALUES (1, 'Ciprofloxacin', 200.00, '2025-11-10', '2025-10-01');
```

Data Output | Messages | Notifications

```
ERROR:  new row for relation "patient_med" violates check constraint "patient_med_check"
Failing row contains (3, 1, Ciprofloxacin, 200.00, 2025-11-10, 2025-10-01).

SQL state: 23514
Detail: Failing row contains (3, 1, Ciprofloxacin, 200.00, 2025-11-10, 2025-10-01).
```

```sql
-- Inverted dates
INSERT INTO patient_med (patient_id, med_name, dose_mg, start_dt, end_dt)
VALUES (1, 'Ciprofloxacin', 200.00, '2025-11-10', '2025-10-01');

---\ Active Databases - Statement-Level Trigger for Bill Totals
-- Tables
CREATE TABLE bill (
    id SERIAL PRIMARY KEY,
    total NUMERIC(12,2) DEFAULT 0
);
```

```sql
CREATE TABLE bill_item (
    bill_id INT REFERENCES bill(id),
    amount NUMERIC(12,2),
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE bill_audit (
    bill_id INT,
    old_total NUMERIC(12,2),
    new_total NUMERIC(12,2),
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 276 msec.
```

```sql
CREATE TABLE bill_audit (
    bill_id INT,
    old_total NUMERIC(12,2),
    new_total NUMERIC(12,2),
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 111 msec.
```

Query | Query History

```
58  -- Trigger function
59  CREATE OR REPLACE FUNCTION recompute_bill_totals()
60  RETURNS TRIGGER AS $$
61  BEGIN
62      UPDATE bill
63      SET total = COALESCE((
64          SELECT SUM(amount) FROM bill_item WHERE bill_id = NEW.bill_id
65      ), 0)
66      WHERE id = NEW.bill_id;
67
68      INSERT INTO bill_audit (bill_id, old_total, new_total)
69      VALUES (
70          NEW.bill_id,
71          0,
72          (SELECT total FROM bill WHERE id = NEW.bill_id)
73      );
74
75      RETURN NULL;
76  END;
77  $$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 239 msec.
```

Total rows: | Query complete 00:00:00.239

```
79  -- Statement-level trigger
80  CREATE TRIGGER trg_bill_total_stmt
81  AFTER INSERT OR UPDATE OR DELETE ON bill_item
82  FOR EACH ROW
83  EXECUTE FUNCTION recompute_bill_totals();
84
```

Data Output | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 139 msec.
```

```
85  ---\ Deductive Databases – Supervision Chain
86  -- Supervision table
87  CREATE TABLE staff_supervisor (
88      employee VARCHAR(50),
89      supervisor VARCHAR(50)
90  );
91
92  -- Recursive query
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 208 msec.
```

Query | Query History

```
91
92  WITH RECURSIVE supers(emp, sup, hops, path) AS (
93      -- Anchor: start with direct supervision links, cast path to TEXT
94      SELECT employee, supervisor, 1, employee::TEXT
95      FROM staff_supervisor
96
97      UNION ALL
98
99      -- Recursive: climb up the supervision chain
100     SELECT s.employee, t.sup, t.hops + 1, t.path || '>' || t.sup
101     FROM staff_supervisor s
102     JOIN supers t ON s.supervisor = t.emp
103     WHERE POSITION(t.sup IN t.path) = 0
104 )
105 -- Final selection: deepest supervisor per employee
106 SELECT emp, sup AS top_supervisor, hops
107 FROM supers
108 WHERE (emp, hops) IN (
109     SELECT emp, MAX(hops) FROM supers GROUP BY emp
110 );
```

Data Output | Messages | Notifications

SQL

emp character varying (50)	top_supervisor character varying (50)	hops integer

Total rows: 0 | Query complete 00:00:00.456

```
112 ---\ Deductive Databases – Supervision Chain
113 -- Supervision table
114 CREATE TABLE staff_supervisor (
115     employee VARCHAR(50),
116     supervisor VARCHAR(50)
117 );
118
119 -- Recursive query
```

Data Output | Messages | Notifications

```
ERROR:  relation "staff_supervisor" already exists

SQL state: 42P07
```

```
112 ---\ Knowledge Bases – Infectious Disease Roll-Up
113 -- Triple table
114 CREATE TABLE triple (
115     s VARCHAR(100),
116     p VARCHAR(50),
117     o VARCHAR(100)
118 );
119
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 309 msec.
```

Total rows: | Query complete 00:00:00.309

Query | Query History

```
119
120  -- Recursive isA closure
121  WITH RECURSIVE isa(entity, category) AS (
122      SELECT s, o FROM triple WHERE p = 'isA'
123      UNION ALL
124      SELECT t.s, i.category
125      FROM triple t
126      JOIN isa i ON t.o = i.entity
127      WHERE t.p = 'isA'
128  ),
129  infectious_patients AS (
130      SELECT DISTINCT t.s
131      FROM triple t
132      JOIN isa ON t.o = isa.entity
133      WHERE t.p = 'hasDiagnosis' AND isa.category = 'InfectiousDisease'
134  )
135  SELECT s AS patient_id FROM infectious_patients;
```

Data Output | Messages | Notifications

SQL

	patient_id character varying (100)

```
13  CREATE EXTENSION postgis;
14  SELECT PostGIS_Version();
15
```

Data Output | Messages | Notifications

```
CREATE EXTENSION

Query returned successfully in 1 secs 591 msec.
```

```
141  SELECT PostGIS_Version();
142
143  ---\ Enable PostGIS Extension
144  CREATE EXTENSION IF NOT EXISTS postgis;
145
146  -- Clinic table
147  CREATE TABLE clinic (
```

Data Output | Messages | Notifications

SQL

	postgis_version text
1	3.5 USE_GEOS=1 USE_PROJ=1 USE_STATS=1

```
144  -- Clinic table
145  CREATE TABLE clinic (
146      id SERIAL PRIMARY KEY,
147      name VARCHAR(100),
148      geom GEOMETRY(Point, 4326)
149  );
150
151  -- Spatial index
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 120 msec.
```

```
151  -- Spatial index
152  CREATE INDEX clinic_spx ON clinic USING GIST (geom);
153
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 82 msec.
```

```sql
-- Ambulance location
SELECT id, name
FROM clinic
WHERE ST_DWithin(
    geom,
    ST_SetSRID(ST_MakePoint(30.0600, -1.9570), 4326),
    0.001   -- ~1km in degrees
);
```

Data Output | Messages | Notifications

SQL

id [PK] integer	name character varying (100)

Query | Query History

```sql
-- Nearest 3 clinics
SELECT id, name,
       ST_Distance(
           geom,
           ST_SetSRID(ST_MakePoint(30.0600,
       ) AS km
FROM clinic
ORDER BY km
LIMIT 3;
```

Data Output | Messages | Notifications

SQL

id [PK] integer	name character varying (100)	km double precision

Query | Query History

```sql
    INTO bill_ids;

    -- Loop through each affected BILL_ID
    FOREACH bill_id IN ARRAY bill_ids LOOP
        SELECT total INTO old_total FROM bill WHERE id = bill_id;

        SELECT COALESCE(SUM(amount), 0)
        INTO new_total
        FROM bill_item
        WHERE bill_id = bill_id;

        UPDATE bill SET total = new_total WHERE id = bill_id;

        INSERT INTO bill_audit (bill_id, old_total, new_total)
        VALUES (bill_id, old_total, new_total);
    END LOOP;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 91 msec.
```

Total rows: | Query complete 00:00:00.091

```sql
-- Statement-level trigger (not FOR EACH ROW)
DROP TRIGGER IF EXISTS trg_bill_total_stmt ON bill_item;

CREATE TRIGGER trg_bill_total_stmt
AFTER INSERT OR UPDATE OR DELETE ON bill_item
FOR EACH STATEMENT
EXECUTE FUNCTION recompute_bill_totals_stmt();
```

Data Output | Messages | Notifications

```
NOTICE:  trigger "trg_bill_total_stmt" for relation "bill_item" does not exist, skipping
CREATE TRIGGER

Query returned successfully in 123 msec.
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_collect_bill_ids
AFTER INSERT OR UPDATE OR DELETE ON bill_item
FOR EACH ROW
EXECUTE FUNCTION collect_bill_ids();

CREATE OR REPLACE FUNCTION recompute_bill_totals_stmt()
RETURNS TRIGGER AS $$
DECLARE
    bill_row RECORD;
    old_total NUMERIC(12,2);
    new_total NUMERIC(12,2);
BEGIN
    FOR bill_row IN SELECT DISTINCT bill_id FROM bill_ids_temp LOOP
        SELECT total INTO old_total FROM bill WHERE id = bill_row.bill_id;

        SELECT COALESCE(SUM(amount), 0)
        INTO new_total
        FROM bill_item
```

Query | Query History

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 76 msec.
```

Total rows: | Query complete 00:00:00.076

```
CREATE TRIGGER trg_recompute_totals_stmt
AFTER INSERT OR UPDATE OR DELETE ON bill_item
FOR EACH STATEMENT
EXECUTE FUNCTION recompute_bill_totals_stmt();
```

Data Output | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 118 msec.
```