SQL CONSTRAINTS & TRIGGERS

1) Check constraint

```
CHECK (conditional-expression)
```

↳ Update/insertion is rejected if the condition does not evaluate to TRUE

→ e.g.
```
CREATE TABLE Invoices (
    invid    INTEGER PRIMARY KEY,
    ordid    INTEGER NOT NULL UNIQUE,
    amount   NUMERIC(8,2) CHECK (amount > 0),
    issued   DATE,
    due      DATE,
    CHECK ( ordid IN SELECT ordid FROM Orders),
    CHECK ( due >= issued));
```

- Can be used inline
- SQL allows queries in CHECK, but this is not implemented in PostgreSQL
- ↳ Similar to foreign key constraint, but not the same:
  - → This check constraint doesn't enforce ordid to be a key for Orders.
  - → This check constraint will fail w/ NULLs

2) Domain constraint

```
CREATE DOMAIN name datatype [DEFAULT value] [constraint]
    where constraint can be NOT NULL or CHECK (expression)
```

↳ A domain is essentially a data type w/ optional constraints

→ e.g.
```
CREATE DOMAIN posnumber NUMERIC(10, 2)
    CHECK ( VALUE > 0 );
```
↳ refers to the value being tested

e.g.
```
CREATE DOMAIN category VARCHAR(20)
    CHECK ( VALUE IN ('BOOK', 'MUSIC', 'MOVIE' ));
```
↳ Value can only be one of these strings

→ After creating the domain, we can use it as follows:

```
CREATE TABLE R (
    a posnumber,
    b category );
```

If we try inserting:
```
INSERT INTO R
VALUES (-1, 'movie')
```
→ Gives an error!

- Value for domain posnumber violates check constraint
- Strings are case sensitive, so value for domain category violates check constraint

→ It behaves like a new data type w/ the additional constraint,

↳ can avoid repeating check constraints

3) Assertions

→ are check constraints that are not bound to a specific table

```
CREATE ASSERTION name CHECK (condition)
```

e.g. CREATE ASSERTION too_many_customers
CHECK ((SELECT COUNT(*) FROM Customers) <= 1000);

↳ Not implemented in any of the currently available DBMS, again the problem is allowing subqueries in CHECK (Not the create assertion)

TRIGGERS

→ specify an action to execute if certain events take place
↓
a change to the database that activates the trigger
(an insertion/deletion/update)

→ 3 components:
- Event
- Condition – a query/test checked when trigger is activated
- Action – a procedure executed when condition is true
  - ↳ can refer to old/new values of modified tuples
  - ↳ can examine answers to the condition query
  - ↳ can execute new queries
  - ↳ can make changes to the database
  - ↳ can be executed before/after the event for each row/for each statement