- use case
  - query result instead of [[SQL]] table
  - aggregation value instead of scalar value
- subqueries can be stored as variable
  - WITH VariableName AS (SELECT ...)
  - Modularization with

    WITH C AS (SELECT …)
- common use cases:
  - **Subqueries w/ IN**
    - Check containment of values in result set of sub query

```
SELECT Product, Quantity, Price
FROM Sales
WHERE Product NOT IN(
   SELECT Product FROM Sales
   GROUP BY Product
   HAVING sum(Quantity*Price)>1e6)
```

  - **Other subqueries**
    - EXISTS: existential quantifier ∃x for correlated subqueries
    - ALL: comparison (w/ universal quantifier ∀x)
    - SOME/ANY: comparison (w/ existential quantifier ∃x)

Correlated and Uncorrelated Subqueries

- correlated if queries depend on each other
  - subquery exectuted for each tuple of outer query
    - nested for loop
  - inefficient

```
SELECT P.Fname, P.Lname
FROM Professors P,
WHERE NOT EXISTS(
    SELECT * FROM Courses C
    WHERE C.PID=P.PID);
```

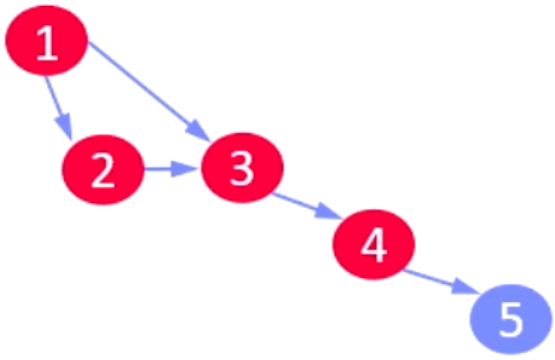
- uncorrelated if subquery executed once

```
SELECT P.Fname, P.Lname
FROM Professors P,
WHERE P.PID NOT IN(
    SELECT PID FROM Courses);
```

- correlated queries may be unnested (de-correlation)
  - can also be improved by "only" executing subquery for each distinct value

Recursive Queries

- terminates when recursive query returns empty table
  - **Approach**
    - `WITH RECURSIVE <name> (<arguments>)`
    - Compose recursive table from **non-recursive term**, **union all/distinct**, and **recursive term**
    - Terminates when recursive term yields empty result
-
  - **Example**
    - Courses(CID, Name), Precond(pre REF CID, suc REF CID)
    - Dependency graph (pre→suc)



-

```
WITH RECURSIVE rPrereq(p,s) AS(
   (SELECT pre, suc
      FROM Precond WHERE suc=5)
  UNION DISTINCT
   (SELECT B.pre, B.suc
    FROM Precond B, rPrereq R
    WHERE B.suc = R.p)
)
SELECT DISTINCT p FROM rPrereq
```

4 3 1 2

-

[[Correlation]]