CS-442 Notes

October 18, 2021

`WITH` is a way to create intermediate tables / temporary tables without actually making another table.

Difference between `VIEW` and `WITH`

- Views are not temporary and are 'public property' meaning anyone can access it.
- Stick to `WITH` for now on any assignments and classwork.
- Both serve the same use case of building intermediate solutions.

Nested subquery = table.

From Chapter 3, Slide 42:

```
SELECT branch_name, avg_balance
FROM (SELECT branch_name, avg(balance)
      FROM account
      GROUP BY branch_name)
      AS branch_avg( branch_name, avg_balance)
WHERE avg_balance > 1200
```

`WITH` equivalent:

```
WITH br_avg AS
(
    SELECT br_name, avg(bal) AS avg_bal
    FROM acct
    GROUP BY br_name

)
SELECT br_name, avg_bal
FROM br_avg
WHERE avg_bal > 1200
```

This is the answer to question 1 on the HW!

Step 1: Get the aggregates

```
WITH agg AS
(
    SELECT cust, min(quantity) AS min_q, max(quantity) as max_q,
avg(quantity) AS avg_g
    GROUP BY cust
)
```

Step 2: Get details for minimum

```
WITH agg AS
(
    SELECT cust, min(quantity) AS min_q, max(quantity) as max_q,
avg(quantity) AS avg_g
    GROUP BY cust
),
min_detail
(
    SELECT a.cust, a.min_q, a.max_q, a.avg_q, s.prod AS min_prod, s.date AS
min_date, s.state AS min_state
    FROM agg AS a, sales AS s
    WHERE a.cust = s.cust
    AND a.min_q = s.q
)
```

Step 3: Final output

```
SELECT m.cust, m.min_q, m.prod, m.date. m.state, m.max_q, m.avg_q, s.prod,
s.date, s.state,
FROM min_detail AS m, sales AS s
WHERE m.cust = s.cust
AND m.max_q = s.quantity
```