CS262- Problem Set 1

CS262- Database Systems
2022-CS-07 — Harmain Iftikhar

February 15, 2024

Problem 1: Products with Cost Higher than Average Cost

Relational Algebra:

$$\pi_{\text{name}}\left(\sigma_{\text{cost}>(\text{SELECT AVG(cost) FROM Product})}\right)(\text{Product})$$

SQL Solutions:
1- Cartesian Product:

```
SELECT name
FROM Product
WHERE cost > (SELECT AVG(p.cost) FROM Product p);
```

2- Joins:

```
SELECT p1.name
FROM Product p1
JOIN (SELECT AVG(cost) AS avg_cost FROM Product) AS avg
ON p1.cost > avg.avg_cost;
```

3- Subquery:

```
SELECT name
FROM Product
WHERE cost > (SELECT AVG(cost) FROM Product);
```

Problem 2: List the name of companies whose products are bought by Aslam

Relational Algebra Expression:

$$\pi_{\text{name}}\left(\sigma_{\text{buyer}='Aslam'}(\text{Purchase} \bowtie \text{Product} \bowtie \text{Company})\right)$$

SQL Solutions:
Cartesian Product:

```
SELECT DISTINCT c.name
FROM Company c, Product p, Purchase pu
WHERE pu.buyer = 'Aslam' AND pu.product = p.name AND p.maker = c.name;
```

Joins:

```
SELECT DISTINCT c.name
FROM Company c
JOIN Product p ON c.name = p.maker
JOIN Purchase pu ON p.name = pu.product
WHERE pu.buyer = 'Aslam';
```

Subquery:

```
SELECT DISTINCT name
FROM Company
WHERE name IN (SELECT maker FROM Product WHERE name IN (SELECT product FROM Purchase WHERE buyer = 'Asl
```

Problem 3: List the name of products that are more expensive than all the products produced by Unilever

Relational Algebra Expression:

$$\pi_{\text{name}}\left(\sigma_{\text{cost}>\text{ALL}(\sigma_{\text{maker}='Unilever'}(\text{Product}))}\right)$$

SQL Solutions:
Cartesian Product:
Not Possible because the ALL operator is not directly supported in SQL Cartesian product. **Joins:**

```
SELECT DISTINCT p1.name
FROM Product p1
JOIN Product p2 ON p1.cost > ALL(SELECT cost FROM Product WHERE maker = 'Unilever');
```

Subquery:

```
SELECT DISTINCT name
FROM Product
WHERE cost > ALL(SELECT cost FROM Product WHERE maker = 'Unilever');
```

Problem 4: List the copycat products along with manufacturer, i.e., the products that have the same name as produced by Unilever

Relational Algebra Expression:

$$\pi_{\text{p1.name, p1.maker}}\,(\sigma_{\text{p1.name=p2.name AND p1.maker ¡¿ p2.maker}}(\text{Product AS p1} \bowtie \text{Product AS p2}))$$

SQL Solutions:
Cartesian Product:
Not Possible because Cartesian product cannot be used to filter out only the copycat products. **Joins:**

```
SELECT DISTINCT p1.name, p1.maker
FROM Product p1
JOIN Product p2 ON p1.name = p2.name AND p1.maker <> p2.maker
WHERE p1.maker = 'Unilever';
```

Subquery:
Not Possible because Subquery alone cannot filter out only the copycat products.

Problem 5: Buyers of products produced in Lahore

Relational Algebra Expression:

$$\pi_{\text{buyer}}\left(\sigma_{\text{city}='Lahore'}(\text{Purchase} \bowtie \text{Product} \bowtie \text{Company})\right)$$

SQL Solutions
Cartesian Product:
Not Possible because Cartesian product does not filter out only the buyers of products produced in Lahore.
Joins:

```
SELECT DISTINCT pu.buyer
FROM Purchase pu
JOIN Product p ON pu.product = p.name
JOIN Company c ON p.maker = c.name
WHERE c.city = ’Lahore’;
```

Subquery:

```
SELECT DISTINCT buyer
FROM Purchase
WHERE product IN (SELECT name FROM Product WHERE maker IN (SELECT name FROM Company WHERE city = ’Lahore
```

Problem 6: List of buyers who only buy the products ’Made in Karachi’

Relational Algebra Expression:

$$\pi_{\text{buyer}}\left(\text{Purchase} - \sigma_{\text{city}<>'Karachi'}(\text{Purchase} \bowtie \text{Product} \bowtie \text{Company})\right)$$

SQL Solutions:
Cartesian Product:
Not Possible because Cartesian product does not facilitate filtering out buyers who only buy products made in Karachi. **Joins:**
Not Possible because Joins alone cannot filter out buyers who only buy products made in Karachi. **Subquery:**

```
SELECT DISTINCT buyer
FROM Purchase
WHERE buyer NOT IN (SELECT buyer FROM Purchase WHERE product IN
                     (SELECT name FROM Product WHERE maker IN
                      (SELECT name FROM Company WHERE city <> ’Karachi’)));
```

Problem 7: Name and price of products bought by more than five customers

Relational Algebra Expression:

$$\pi_{\text{product, price}}\left(\sigma_{\text{count(buyer) ¿ 5}}(\text{Purchase})\right)$$

SQL Solutions:
Cartesian Product:
Not Possible because Cartesian product does not facilitate counting the number of buyers for each product.
Joins:
Not Possible because Joins alone cannot count the number of buyers for each product. **Subquery:**

```
SELECT product, price
FROM Purchase
GROUP BY product, price
HAVING COUNT(buyer) > 5;
```

Problem 8: List of products that are more expensive than all the products made by the same company before 2015

Relational Algebra Expression:

$$\pi_{\text{name}}\left(\sigma_{\text{cost}>\text{ALL}(\sigma_{\text{year}<2015\wedge\text{maker}=p1.maker}(\text{Product AS p1}))}\right)$$

SQL Solutions:
Cartesian Product:
Not Possible because Cartesian product does not facilitate filtering out products based on their launch year.
Joins:

```
SELECT DISTINCT p1.name
FROM Product p1
JOIN Product p2 ON p1.maker = p2.maker AND p1.cost > p2.cost
WHERE p2.year < 2015;
```

Subquery:

```
SELECT DISTINCT name
FROM Product
WHERE cost > ALL(SELECT cost FROM Product WHERE maker = p1.maker AND year < 2015);
```

Problem 9: List of companies who never sell products with loss

Relational Algebra Expression:

$$\pi_{\text{name}}\left(\text{Company}-\pi_{\text{maker}}\left(\sigma_{\text{price}<\text{cost}}(\text{Purchase}\bowtie\text{Product}\bowtie\text{Company})\right)\right)$$

SQL Solutions:
Cartesian Product:
Not Possible because Cartesian product does not facilitate filtering out companies based on whether they ever sold products with loss. **Joins:**
Not Possible because Joins alone cannot filter out companies who never sold products with loss. **Subquery:**

```
SELECT name
FROM Company
WHERE name NOT IN (SELECT maker FROM Purchase WHERE price <
                   (SELECT cost FROM Product WHERE Product.name = Purchase.product));
```

Problem 10: List the products which have more than average revenue in 2015 but below average revenue in 2016

Relational Algebra Expression:

$$\pi_{\text{product}}\left(\sigma_{\text{revenue}>\text{AVG(revenue\_2015)}\wedge\text{revenue}<\text{AVG(revenue\_2016)}}(\text{Product}\bowtie\text{Purchase})\right)$$

SQL Solutions:
Cartesian Product:
Not Possible because Cartesian product does not directly support calculations of average revenue for different years.