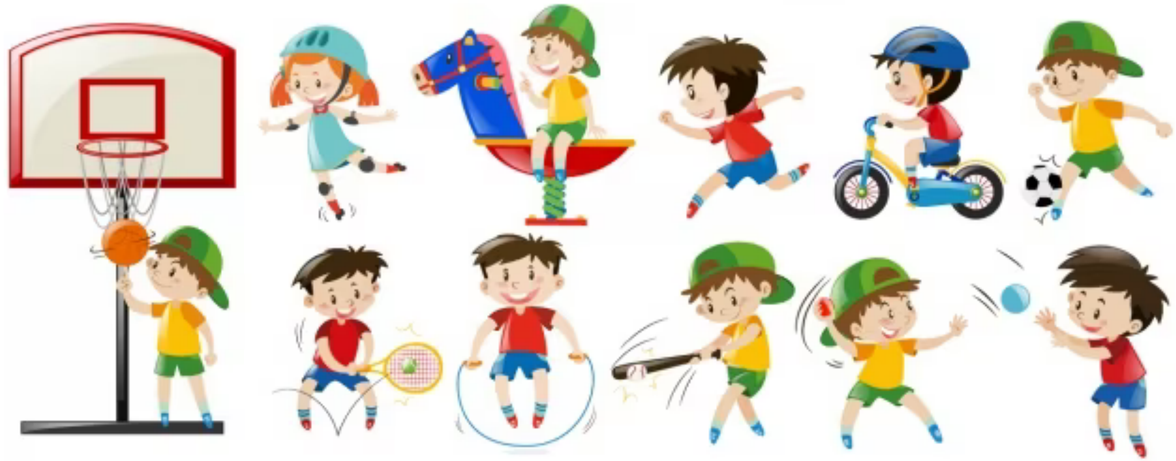
SubQuery

SQL Query Within A Query

Database for Software Developers

Md. Emran Hasan
Anis Uddin Ahmad

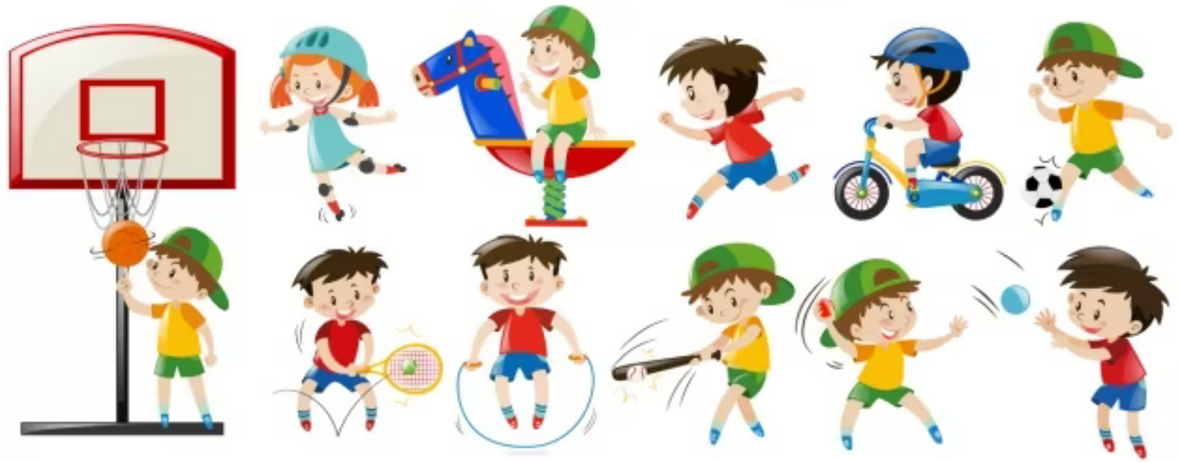
Make a list of students of class 4 who attended in annual sports.



Condition 1 → Make a list of students

Condition 2 → of class 4

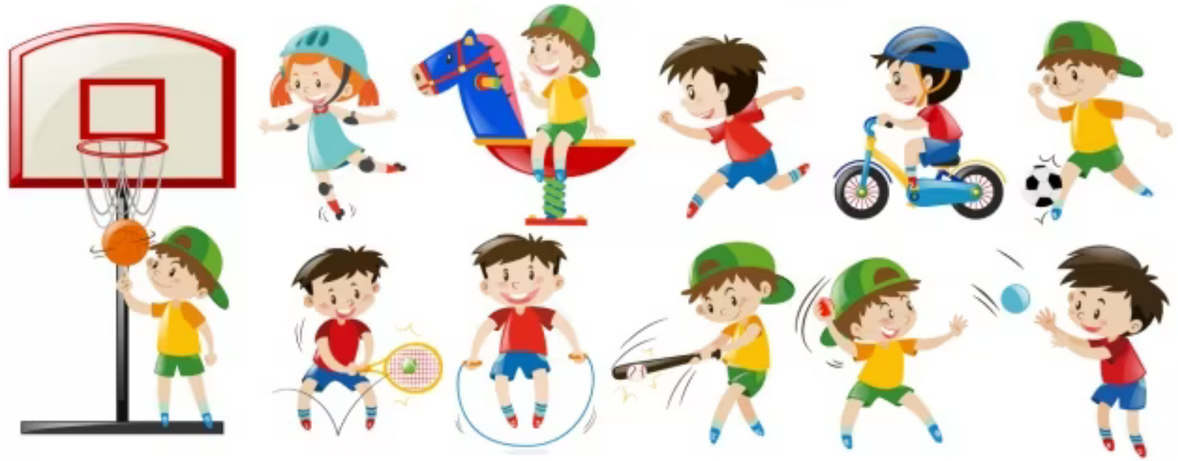
who attended in annual sports.



Condition 1 → Make a list of students of class 4

Condition 2 → who attended in annual sports.

Make a list of participants



Subquery

Subquery

Query inside a Query

Eliminating or including records
based on the results of a secondary query

SubQuery - Filtering Results

Subquery can be used to filter results based on a condition.

```
-- Get list of customers
-- who took rent at least one film
-- in a given month
```

SubQuery - Filtering Results

Subquery can be used to filter results based on a condition.

```
-- Get list of customers
-- who took rent at least one film
-- in a given month
SELECT first_name, last_name, email FROM customer
;
```

SubQuery - Filtering Results

Subquery can be used to filter results based on a condition.

```
-- Get list of customers
-- who took rent at least one film
-- in a given month
SELECT first_name, last_name, email FROM customer
WHERE customer_id IN (
    SELECT customer_id FROM rental
);
```

SubQuery - Filtering Results

Subquery can be used to filter results based on a condition.

```sql
-- Get list of customers
-- who took rent at least one film
-- in a given month
SELECT first_name, last_name, email FROM customer
WHERE customer_id IN (
    SELECT customer_id FROM rental
    WHERE rental_date BETWEEN '2005-05-01' AND '2005-05-31'
);
```

SubQuery - Dependent aggregation

Subquery can be used to get aggregated values

that are connected with current record

```
-- Get list of actors with their total films
SELECT actor_id, first_name, last_name, (
    SELECT COUNT(*) FROM film_actor
    WHERE actor_id = actor.actor_id
    ) AS total_films
FROM actor;
```

SubQuery - Dependent aggregation

Subquery can be used to get aggregated values

that are connected with current record

```
-- Get list of actors with their total films
SELECT actor_id, first_name, last_name, (
    SELECT COUNT(*) FROM film_actor
    WHERE actor_id = actor.actor_id
    ) AS total_films
FROM actor;
```

Correlated subquery

SubQuery - Filtering by aggregated result

Subquery can be used to compare with aggregated values

```
-- Get list of actors who have more than 30 films
SELECT actor_id, first_name, last_name, (
    SELECT COUNT(*) FROM film_actor
    WHERE actor_id = actor.actor_id
    ) AS total_films
FROM actor
HAVING total_films > 30;
```

SubQuery - Filtering by Existence

Subquery can be used to filter records

based on existence of records in a related dataset

```
-- Find the customers who have not returned an inventory
SELECT customer_id, first_name, last_name, email
FROM customer WHERE EXISTS (
    SELECT * FROM rental
    WHERE rental.customer_id = customer.customer_id
        AND return_date IS NULL
    );
```

SubQuery - Filtering by Existence

Subquery can be used to filter records

based on existence of records in a related dataset

```
-- Find the customers who have ALREADY returned an inventory
SELECT customer_id, first_name, last_name, email
FROM customer WHERE NOT EXISTS (
    SELECT * FROM rental
    WHERE rental.customer_id = customer.customer_id
        AND return_date IS NULL
    );
```

SubQuery - Nested Queries

Subquery can be used within a subquery

```
-- Average replacement_cost per category
SELECT c.category_id, c.name, (
    SELECT AVG(replacement_cost) FROM film f
    WHERE f.film_id IN (
        SELECT film_id FROM film_category fc
        WHERE fc.category_id = c.category_id
        )
    ) AS avg_replacement_cost
FROM category c;
```

SubQuery - Nested Queries

Subquery can be used within a subquery

```
-- Average replacement_cost per category
SELECT c.category_id, c.name, (
    SELECT AVG(replacement_cost) FROM film f
    WHERE f.film_id IN (
        SELECT film_id FROM film_category fc
        WHERE fc.category_id = c.category_id
        )
    ) AS avg_replacement_cost
FROM category c;
```

SubQuery - Nested Queries

Subquery can be used within a subquery

```
-- Average replacement_cost per category
SELECT c.category_id, c.name, (
    SELECT AVG(replacement_cost) FROM film f
    WHERE f.film_id IN (
        SELECT film_id FROM film_category fc
        WHERE fc.category_id = c.category_id
        )
    ) AS avg_replacement_cost
FROM category c;
```

SubQuery - Nested Queries

Subquery can be used within a subquery

```
-- Average replacement_cost per category
SELECT c.category_id, c.name, (
    SELECT AVG(replacement_cost) FROM film f
    WHERE f.film_id IN (
        SELECT film_id FROM film_category fc
        WHERE fc.category_id = c.category_id
        )
    ) AS avg_replacement_cost
FROM category c;
```

SubQuery - Nested Queries

Subquery can be used within a subquery

```
-- Average replacement_cost per category
SELECT c.category_id, c.name, (
    SELECT AVG(replacement_cost) FROM film f
    WHERE f.film_id IN (
        SELECT film_id FROM film_category fc
        WHERE fc.category_id = c.category_id
        )
    ) AS avg_replacement_cost
FROM category c;
```

SubQuery - Nested Queries

Subquery can be used within a subquery

```
-- Average replacement_cost per category
SELECT c.category_id, c.name, (
    SELECT AVG(replacement_cost) FROM film f
    WHERE f.film_id IN (
        SELECT film_id FROM film_category fc
        WHERE fc.category_id = c.category_id
        )
    ) AS avg_replacement_cost
FROM category c;
```

SubQuery - Selecting single values

Subquery can be used to select a single or aggregated value from the same or a different table.

```
-- Get total number of inventories
-- and number of inventories that are not yet returned
SELECT
    COUNT(*) total_inventory,
    (SELECT COUNT(*) FROM rental WHERE return_date IS NULL) yet_to_return
FROM inventory;
```

SubQuery for UPDATE/DELETE

Subquery can be used to filter records in UPDATE or DELETE.

```
-- Update replacement_cost of films under a specific category
UPDATE film
SET replacement_cost = replacement_cost * 1.1
WHERE film_id IN (
    SELECT film_id FROM film_category
    WHERE category_id = 1
    );
```

SubQuery VS Multiple Query

```sql
-- Get list of customers who took rent at least one film in a given month
SELECT first_name, last_name, email FROM customer
WHERE customer_id IN (
    SELECT customer_id FROM rental
    WHERE rental_date BETWEEN '2005-05-01' AND '2005-05-31'
);
```

VS

```sql
-- 1. Get list of customer ids who took rent at least one film in a given month
SELECT customer_id FROM rental
WHERE rental_date BETWEEN '2005-05-01' AND '2005-05-31'

-- 2. Get additional details of customer by retrieved IDs
SELECT first_name, last_name, email FROM customer
WHERE customer_id IN (2, 4, 56, 67, 78, 88, 89,...);
```

SubQuery VS Multiple Query

Subquery:

- ☑ Will be benefitted from MySQL's internal optimization
- ☑ Involves a single cycle of

```
client → server → parsing → execution → return
```

Multiple Query:

- ☑ Simple to write and understand
- ☑ Easy to identify and fix issues

SubQuery VS Multiple Query

1. Breakdown, prepare and test as Multiple-Queries
2. Combine them into a Subquery

SubQuery - When to consider?

- ☑ Not Interested in columns from join tables
- ☑ If no direct connection between tables to join together
- ☑ Need to compare a single value or a result of an aggregation
- ☑ To avoid long chain of connecting joins to reach a distant filtering column
- ☑ To gather disconnected aggregations with single query
- ☑ Check and balance - compare the performance impact

Questions?