

SQL Subqueries

টেবিল ডেটা:

Students Table:

	student_id	student_name	class	age
1		Alice	4DM	20
2		Bob	4DM	21
3		Charlie	4DM	19
4		David	4DM	22

Courses Table:

	course_id	course_name	student_id	marks
101		Database	1	85
102		Math	1	90
103		Physics	2	75
104		Chemistry	3	88

Departments Table:

	dept_id	dept_name	budget
1		Science	50000
2		Arts	30000
3		Commerce	40000

1. Single-row Subquery (WHERE clause-এ)

কোড়:

```
sql|
```

```
SELECT student_name, age  
FROM students  
WHERE age > (SELECT AVG(age) FROM students);
```

ব্যাখ্যা:

- Subquery প্রথমে execute হয় এবং একটি single value return করে
- এখানে AVG(age) সব students-দের average age বের করে
- Main query সেই average-এর চেয়ে বেশি age-এর students খুঁজে বের করে
- Subquery একবার execute হয় এবং result main query-তে use হয়

আউটপুট:

student_name	age
Bob	21
David	22

মনে রাখার কথা: একটি value return করে, comparison-এ ব্যবহার হয়

2. Multiple-row Subquery (IN operator)

কোড:

```
sql
```

```
SELECT student_name
```

```
FROM students
```

```
WHERE student_id IN (SELECT student_id FROM courses WHERE marks > 80);
```

ব্যাখ্যা:

- Subquery multiple values return করে (একাধিক student_id)
- IN operator দিয়ে check করা হয় main query-র value, subquery-র results-এ আছে কিনা
- এখানে যে students 80-এর বেশি marks পেয়েছে তাদের খুঁজছি

- Subquery একাধিক rows return করতে পারে

আউটপুট:

student_name

Alice

Charlie

মনে রাখার কথা: একাধিক value return করে, IN/NOT IN-এ ব্যবহার হয়

3. Correlated Subquery

কোড:

sql

```
SELECT s.student_name, s.age
FROM students s
WHERE s.age > (SELECT AVG(age) FROM students WHERE class = s.class);
```

ব্যাখ্যা:

- Correlated Subquery বাইরের query-র প্রতিটি row-এর জন্য আলাদাভাবে execute হয়
- Inner query, outer query-র column (s.class) ব্যবহার করে
- প্রতিটি student-এর জন্য তার নিজের class-এর average age check করে
- Performance-এ একটু slow হতে পারে কারণ বারবার execute হয়

আউটপুট:

student_name age

Bob	21
-----	----

David	22
-------	----

মনে রাখার কথা: বাইরের query-র সাথে সম্পর্কিত, বারবার execute হয়

4. Subquery in SELECT clause

কোড:

sql

```
SELECT student_name,  
       (SELECT COUNT(*) FROM courses c WHERE c.student_id = s.student_id) AS total_courses  
FROM students s;
```

ব্যাখ্যা:

- SELECT clause-এ subquery ব্যবহার করে প্রতিটি row-এর জন্য calculated value আনা
- প্রতিটি student-এর ক্ষেত্রে courses আছে সেটা count করছে
- প্রতিটি row-এর জন্য subquery আলাদাভাবে run হয়
- এটাও একটি correlated subquery

আউটপুট:

student_name total_courses

Alice	2
Bob	1
Charlie	1
David	0

মনে রাখার কথা: প্রতিটি row-এর জন্য calculated column তৈরি করে

5. Subquery in FROM clause (Derived Table)

কোড:

sql

```
SELECT avg_marks_table.student_id, avg_marks_table.avg_marks  
FROM (SELECT student_id, AVG(marks) AS avg_marks
```

```
FROM courses  
GROUP BY student_id) AS avg_marks_table  
WHERE avg_marks_table.avg_marks > 80;
```

ব্যাখ্যা:

- FROM clause-এ subquery একটি temporary table (derived table) তৈরি করে
- এই temporary table-কে alias দিতে হয় (এখানে avg_marks_table)
- Subquery-র result যেন একটি নতুন table, যেটা থেকে data select করা হচ্ছে
- Complex calculations করে তারপর সেই result থেকে filter করতে সুবিধা

আউটপুট:

student_id	avg_marks
1	87.5
3	88.0

মনে রাখার কথা: Temporary table তৈরি করে, alias বাধ্যতামূলক

6. Subquery with EXISTS

কোড:

```
sql  
SELECT student_name  
FROM students s  
WHERE EXISTS (SELECT 1 FROM courses c WHERE c.student_id = s.student_id AND c.marks >  
85);
```

ব্যাখ্যা:

- EXISTS check করে subquery কোনো row return করে কিনা
- Row থাকলে TRUE, না থাকলে FALSE
- এখানে যে students-দের কোনো course-এ 85-এর বেশি marks আছে তাদের খুঁজছি

- EXISTS শব্দু �existence check করে, actual data return করে না
- Performance-এ দ্রুত কারণ match পেলেই থেমে যায়

আউটপুট:

student_name

Alice

Charlie

মনে রাখার কথা: EXISTS existence check করে, দ্রুত performance

7. Subquery with NOT EXISTS

কোড:

sql

```
SELECT student_name
FROM students s
WHERE NOT EXISTS (SELECT 1 FROM courses c WHERE c.student_id = s.student_id);
```

ব্যাখ্যা:

- NOT EXISTS check করে subquery কোনো row return করে না কিনা
- যে students-দের কোনো course নেই তাদের খুঁজে বের করে
- ANTI JOIN-এর মতো কাজ করে
- এটা LEFT JOIN ... WHERE column IS NULL-এর alternative

আউটপুট:

student_name

David

মনে রাখার কথা: যাদের match নেই তাদের খুঁজে বের করে

8. Subquery with ALL operator

কোড:

sql

```
SELECT student_name, age
```

```
FROM students
```

```
WHERE age > ALL (SELECT age FROM students WHERE class = '4DM' AND age < 21);
```

ব্যাখ্যা:

- ALL operator subquery-র সব values-এর সাথে compare করে
- Condition সব values-এর জন্য TRUE হতে হবে
- এখানে 21-এর কম সব age-এর চেয়ে বেশি age খুঁজছি
- > ALL মানে সবচেয়ে বড় value-এর চেয়েও বড়

আউটপুট:

student_name	age
Bob	21
David	22

মনে রাখার কথা: সব values satisfy করতে হবে

9. Subquery with ANY/SOME operator

কোড:

sql

```
SELECT student_name, age
```

```
FROM students
```

```
WHERE age > ANY (SELECT age FROM students WHERE class = '4DM' AND age > 20);
```

ব্যাখ্যা:

- ANY/SOME operator subquery-র যেকোনো একটি value satisfy করলেই হয়
- > ANY মানে কমপক্ষে একটি value-এর চেয়ে বড়
- SOME এবং ANY একই জিনিস
- এখানে 20-এর বেশ ages-এর মধ্যে যেকোনো একটির চেয়ে বড় age খুঁজছি

আউটপুট:

student_name age

David 22

মনে রাখার কথা: অন্তত একটি value satisfy করলেই চলবে

10. Nested Subquery (Subquery-র ভিতরে Subquery)

কোড:

```
sql
SELECT student_name
FROM students
WHERE student_id IN (
    SELECT student_id
    FROM courses
    WHERE marks > (
        SELECT AVG(marks)
        FROM courses
    )
);
```

ব্যাখ্যা:

- একের ভিতরে এক subquery থাকতে পারে

- সবচেয়ে ভিতরের subquery আগে execute হয়
- এখানে প্রথমে average marks-কে করা হয়, তারপর সেই average-এর বেশি marks-এর students খুঁজছি
- 3 level পর্যন্ত nest করা যায়, তবে বেশি nest করলে readability কমে যায়

আউটপুট:

student_name

Alice

Charlie

মনে রাখার কথা: ভিতর থেকে বাইরের দিকে execute হয়

11. Subquery with INSERT

কোড:

sql

```
INSERT INTO students (student_id, student_name, class, age)
SELECT student_id + 10, student_name, class, age
FROM students
WHERE age > 20;
```

ব্যাখ্যা:

- INSERT statement-এ subquery ব্যবহার করে একসাথে multiple rows insert করা
- Subquery-র result direct insert হয়
- VALUES clause-এর বদলে SELECT statement ব্যবহার হয়
- এক table থেকে data নিয়ে অন্য table-এ insert করতে সুবিধা

মনে রাখার কথা: Bulk insert করতে ব্যবহার হয়

12. Subquery with UPDATE

কোড:

sql

UPDATE courses

SET marks = marks + 5

WHERE student_id IN (SELECT student_id FROM students WHERE age < 20);

ব্যাখ্যা:

- UPDATE statement-এ subquery ব্যবহার করে conditional update করা
- যে students-দের age 20-এর কম তাদের marks 5 বাড়ানো হচ্ছে
- Subquery dynamically যে rows update হবে সেগুলো select করে
- Complex conditions apply করতে সুবিধা

মনে রাখার কথা: Dynamic condition দিয়ে update করা

13. Subquery with DELETE

কোড:

sql

DELETE FROM courses

WHERE student_id IN (SELECT student_id FROM students WHERE age > 22);

ব্যাখ্যা:

- DELETE statement-এ subquery ব্যবহার করে conditional delete করা
- যে students-দের age 22-এর বেশি তাদের সব courses delete করা হচ্ছে
- Subquery-র result অনুযায়ী rows delete হয়
- এক table-এর data দেখে অন্য table থেকে delete করতে পারি

মনে রাখার কথা: Related data delete করতে ব্যবহার হয়

14. Subquery with HAVING clause

কোড:

```
sql  
SELECT student_id, AVG(marks) AS avg_marks  
FROM courses  
GROUP BY student_id  
HAVING AVG(marks) > (SELECT AVG(marks) FROM courses);
```

ব্যাখ্যা:

- HAVING clause-এ subquery ব্যবহার করে grouped data filter করা
- WHERE clause group হওয়ার আগে কাজ করে, HAVING group হওয়ার পরে
- এখানে যে students-দের average marks, overall average-এর চেয়ে বেশি তাদের খুঁজছি
- Aggregate functions-এর result filter করতে HAVING + Subquery ভালো

আউটপুট:

student_id avg_marks

1	87.5
3	88.0

মনে রাখার কথা: Grouped data filter করতে ব্যবহার হয়

15. Scalar Subquery (Single value return)

কোড:

```
sql  
SELECT student_name, age,  
(SELECT MAX(marks) FROM courses) AS highest_mark_overall  
FROM students;
```

ব্যাখ্যা:

- Scalar subquery ঠিক একটি row এবং একটি column return করে (single value)
- প্রতিটি row-এর জন্য same value দেখাবে
- এখানে সব courses-এর মধ্যে সর্বোচ্চ marks প্রতিটি student-এর সাথে দেখাগো হচ্ছে
- Non-correlated হলে শুধু একবার execute হয়

আউটপুট:

student_name age highest_mark_overall

Alice 20 90

Bob 21 90

Charlie 19 90

David 22 90

মনে রাখার কথা: শুধু একটি value return করে

Subquery তুলনা চার্ট:

Subquery Type	Return করে	কোথায় ব্যবহার হয়	Performance
Single-row	একটি value	WHERE, HAVING	দ্রুত
Multiple-row	একাধিক values	IN, ANY, ALL	মাঝারি
Correlated	প্রতিটি row-এর জন্য WHERE, SELECT		ধীর
Derived Table	পুরো table	FROM	মাঝারি
EXISTS	TRUE/FALSE	WHERE	দ্রুত
Scalar	একটি value	SELECT, WHERE	দ্রুত

কখন কোন Subquery ব্যবহার করবেন?

- **Single-row Subquery:** যখন একটি মাত্র value-র সাথে compare করতে হবে (AVG, MAX, MIN)
 - **Multiple-row Subquery:** যখন একাধিক values-এর মধ্যে check করতে হবে (IN operator)
 - **Correlated Subquery:** যখন বাইরের query-র প্রতিটি row-র জন্য আলাদা calculation লাগবে
 - **Derived Table:** যখন complex calculation করে তারপর সেই result থেকে query করতে হবে
 - **EXISTS:** যখন শুধু existence check করতে হবে, actual data লাগবে না
 - **Scalar Subquery:** যখন একটি calculated value প্রতিটি row-এ দেখাতে হবে
 - **Nested Subquery:** যখন multi-step filtering করতে হবে
-

Performance Tips:

✓ ভালো Practice:

- EXISTS ব্যবহার করুন IN-এর বদলে (large dataset-এ)
- JOIN ব্যবহার করুন correlated subquery-র বদলে যখন সম্ভব
- Subquery-তে index থাকা columns ব্যবহার করুন

✗ এড়িয়ে চলুন:

- SELECT clause-এ অপ্রয়োজনীয় correlated subquery
- বেশি level-এর nested subquery (3+ levels)
- Subquery-তে SELECT * ব্যবহার (শুধু প্রয়োজনীয় columns নিন)