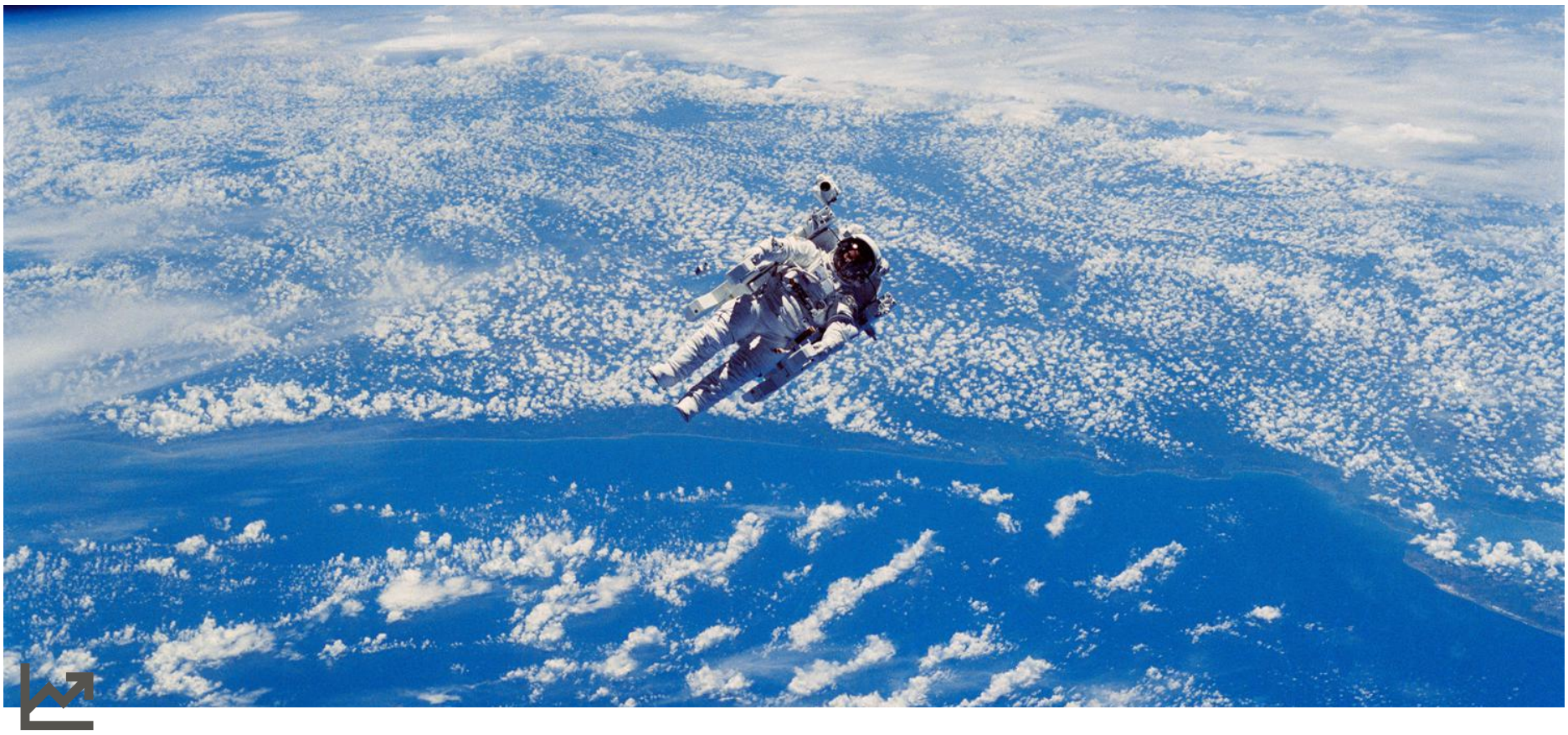

Data Retrieval by Alfan Nurrohim

> 💡 1. Construct and analyze queries that select data
> 2. Construct and analyze queries that sort and filter data
> 3. Construct and analyze queries that aggregate data

Data Retrieval

Data Retrieval adalah proses mengambil atau mengakses data dari sumber penyimpanan data, seperti basis data, file, sumber data eksternal, atau sumber data lainnya. Tujuan utama dari data retrieval adalah untuk mengambil informasi yang diperlukan dari data yang tersimpan sehingga dapat digunakan untuk analisis, pemrosesan, pelaporan, atau tujuan lainnya.

---

1. Construct and analyze queries that select data

INNER JOIN, LEFT JOIN, RIGHT JOIN, CROSS JOIN (Cartesian product), and FULL OUTER JOIN; self joins; combine result sets by using UNION and INTERSECT; DISTINCT; column alias; computed columns

INNER JOIN:

- INNER JOIN menggabungkan baris dari dua tabel berdasarkan kondisi yang diberikan dan hanya menghasilkan baris yang memiliki pasangan yang sesuai dalam kedua tabel.
- INNER JOIN adalah mekanisme JOIN secara default di MySQL, dimana jika terdapat relasi antara tabel pertama dan tabel kedua, maka hasil akan ditampilkan. Jika ada data di tabel pertama yang tidak memiliki relasi di table kedua ataupun sebaliknya, maka hasil tidak akan ditampilkan.

```
SELECT customers.customer_name, orders.order_date
FROM customers
INNER JOIN orders ON customers.customer_id = orders.customer_id;
```

LEFT JOIN:

- LEFT JOIN menggabungkan semua baris dari tabel kiri (tabel pertama) dengan baris yang cocok dari tabel kanan (tabel kedua).

LEFT JOIN adalah mekanisme join seperti INNER JOIN, namun perbedaannya adalah semua data di tabel pertama akan diambil, dan jika ada data yang tidak memiliki relasi di table kedua, maka hasilnya akan NULL. Pada prakteknya LEFT JOIN ini lebih sering digunakan dibandingkan jenis join lainnya.

```
SELECT customers.customer_name, orders.order_date
FROM customers
LEFT JOIN orders ON customers.customer_id = orders.customer_id;
```

RIGHT JOIN:

- RIGHT JOIN adalah kebalikan dari LEFT JOIN. Ini menggabungkan semua baris dari tabel kanan dengan baris yang cocok dari tabel kiri.
- Jika tidak ada pasangan yang sesuai di tabel kiri, baris dari tabel kanan tetap akan dimasukkan dalam hasil akhir dengan nilai NULL untuk kolom dari tabel kiri.

```
SELECT customers.customer_name, orders.order_date
FROM customers
RIGHT JOIN orders ON customers.customer_id = orders.customer_id;
```

CROSS JOIN adalah salah satu JOIN yang sangat jarang sekali digunakan. Dimana dalam mekanismenya, crossjoin akan mengalikan data di tabel pertama dengan data pada tabel kedua.

```
select * from categories
cross join products
```

Self Joins:

- Self join adalah operasi penggabungan di mana tabel digabungkan dengan dirinya sendiri.
- Ini digunakan ketika Anda memiliki data hierarkis atau struktur yang menghubungkan baris dalam tabel dengan baris lain dalam tabel yang sama.

In the previous tutorials, you have learned how to join a table to the other tables using `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, or `CROSS JOIN` clause. However, there is a special case that you need to join a table to itself, which is known as a self join.

```
SELECT
    IFNULL(CONCAT(m.lastname, ', ', m.firstname),
            'Top Manager') AS 'Manager',
    CONCAT(e.lastname, ', ', e.firstname) AS 'Direct report'
FROM
    employees e
LEFT JOIN employees m ON
    m.employeeNumber = e.reportsto
ORDER BY
    manager DESC;
```

DISTINCT:

- DISTINCT digunakan untuk mengembalikan hasil yang unik (tanpa duplikat) dari sebuah SELECT statement.
- Ini berguna ketika Anda ingin menghilangkan duplikat dari hasil query.

```
SELECT DISTINCT department FROM employees;
```

UNION

UNION hampir sama seperti INNER JOIN, hanya saja perbedaannya, union bukan menggabungkan tabel berdasarkan ID secara side by side, tetapi menggabungkan tabel secara vertikal. Oleh karena itu, penggunaan union ini harus memenuhi beberapa peraturan seperti jumlah kolom, tipe kolom, penamaan, dsb.

```
select distinct email from customers
union
select distinct email from guestbooks;
```

UNION ALL

UNION ALL hampir sama seperti UNION, namun jika pada union biasa data duplikat secara otomatis akan dihilangkan, pada union all ini data duplikat akan tetap akan ditampilkan di hasil query nya. Sama seperti union, penggunaan union all ini juga harus memenuhi beberapa peraturan yang sama, seperti jumlah kolom, tipe kolom, penamaan, dsb.

```
select distinct email from customers
union all
select distinct email from guestbooks;
```

INTERSECT

INTERSECT adalah operasi menggabungkan dua query, namun yang diambil hanya data yang terdapat pada hasil query pertama dan query kedua. Data yang tidak hanya ada di salah satu query, kan dihapus di hasil operasi ini (data muncul tidak dalam duplikat).

Sayangnya, MySQL tidak memiliki operator intersect, dengan demikian maka kita harus lakukan secara manual menggunakan INNER JOIN

```
-- melakukan intersect manual dengan inner join
select distinct customers.email from customers
inner join guestbooks on (guesbooks.email = customers.email);
```

2. Construct and analyze queries that sort and filter data

ORDER BY, WHERE, LIKE, BETWEEN, AND, OR, NOT, TOP (LIMIT), IN, NOT IN, ANY, ALL, NULL, NOT NULL, comparison operators

```
-- ORDER BY:
SELECT * FROM products
ORDER BY price DESC;
Hasil query ini akan mengambil semua produk dari tabel products dan mengurutkannya berdasarkan h

-- WHERE:
SELECT * FROM employees
WHERE department = 'Sales';
Query ini akan mengambil semua data pegawai yang bekerja di departemen "Sales".

-- LIKE:
SELECT * FROM customers
WHERE last_name LIKE 'Sm%';
Query ini akan mengambil semua pelanggan yang memiliki nama belakang yang dimulai dengan "Sm", s

-- BETWEEN:
SELECT * FROM orders
WHERE order_date BETWEEN '2023-01-01' AND '2023-03-31';
Ini akan mengambil semua pesanan yang ditempatkan antara 1 Januari 2023 dan 31 Maret 2023.

-- AND:
SELECT * FROM products
WHERE price > 50 AND stock_quantity > 0;
Query ini akan mengambil produk yang harganya lebih dari 50 dan memiliki jumlah stok yang lebih

-- OR:
SELECT * FROM employees
WHERE department = 'Sales' OR department = 'Marketing';
Ini akan mengambil semua pegawai yang bekerja di departemen "Sales" atau "Marketing".

-- NOT:
SELECT * FROM customers
```

```
WHERE NOT country = 'USA';
Query ini akan mengambil semua pelanggan yang bukan berasal dari Amerika Serikat.

-- TOP (LIMIT):
SELECT TOP 5 * FROM products;
Ini akan mengambil lima produk pertama dari tabel products.

-- IN:
SELECT * FROM orders
WHERE customer_id IN (1, 2, 3);
Query ini akan mengambil semua pesanan yang dibuat oleh pelanggan dengan ID 1, 2, atau 3.

-- NOT IN:
SELECT * FROM products
WHERE category_id NOT IN (4, 5);
Ini akan mengambil semua produk yang tidak termasuk dalam kategori 4 atau 5.

-- NULL dan NOT NULL:
SELECT * FROM employees
WHERE manager_id IS NULL;
Ini akan mengambil semua pegawai yang tidak memiliki manajer.

-- Operator perbandingan:
SELECT * FROM students
WHERE age > 18;
Ini akan mengambil semua siswa yang berusia lebih dari 18 tahun.
```

1. `ORDER BY`: Digunakan dalam pernyataan SQL untuk mengurutkan hasil query berdasarkan satu atau beberapa kolom dalam urutan tertentu, seperti ascending (ASC) atau descending (DESC).
2. `WHERE`: Digunakan untuk memfilter hasil query berdasarkan suatu kondisi tertentu. Hanya baris yang memenuhi kondisi tersebut yang akan dimasukkan ke dalam hasil.
3. `LIKE`: Digunakan dalam pernyataan SQL untuk mencocokkan nilai kolom dengan pola teks tertentu. Anda dapat menggunakan wildcard seperti `%` (cocokkan beberapa karakter) atau `_` (cocokkan satu karakter) dengan `LIKE`.
4. `BETWEEN`: Digunakan untuk memfilter nilai kolom yang berada dalam rentang tertentu. Misalnya, `BETWEEN 1 AND 10` akan mencocokkan nilai yang antara 1 hingga 10.
5. `AND`: Digunakan dalam pernyataan SQL untuk mengkombinasikan beberapa kondisi, di mana semua kondisi harus benar agar baris tersebut termasuk dalam hasil.
6. `OR`: Digunakan untuk mengkombinasikan beberapa kondisi, di mana salah satu kondisi harus benar agar baris tersebut termasuk dalam hasil.
7. `NOT`: Digunakan untuk membalikkan hasil dari suatu kondisi. Misalnya, `NOT LIKE 'abc'` akan mencocokkan semua nilai yang tidak sama dengan 'abc'.
8. `TOP` (atau `LIMIT` di beberapa sistem database): Digunakan untuk membatasi jumlah baris yang dikembalikan oleh query. Ini berguna ketika Anda hanya ingin mengambil sejumlah tertentu dari hasil query.
9. `IN`: Digunakan untuk memeriksa apakah nilai suatu kolom cocok dengan salah satu nilai dalam daftar nilai yang ditentukan.
10. `NOT IN`: Kebalikan dari `IN`, digunakan untuk memeriksa apakah nilai suatu kolom tidak cocok dengan salah satu nilai dalam daftar nilai yang ditentukan.
11. `ANY` dan `ALL`: Biasanya digunakan dalam kombinasi dengan subquery. `ANY` memeriksa apakah setidaknya satu hasil subquery cocok dengan kondisi yang diberikan, sedangkan `ALL` memeriksa apakah semua hasil subquery cocok dengan kondisi yang diberikan.
12. `NULL`: Digunakan untuk mewakili nilai yang tidak ada atau tidak diketahui dalam kolom database.
13. `NOT NULL`: Digunakan untuk memeriksa apakah kolom memiliki nilai yang tidak null, artinya nilai yang ada atau diketahui.

14. Operator perbandingan: Ini termasuk operator seperti `=`, `<`, `>`, `<=`, `>=`, dan `!=` yang digunakan untuk membandingkan nilai di dalam kondisi WHERE atau saat membandingkan kolom dalam query SQL.

3. Construct and analyze queries that aggregate data

GROUP BY, HAVING, MIN, MAX, COUNT, AVG (AVERAGE), SUM

1. **GROUP BY**:
   - Digunakan untuk mengelompokkan hasil query berdasarkan satu atau beberapa kolom.
   - Ini memungkinkan Anda untuk melakukan operasi agregasi pada setiap kelompok data terpisah.

   Contoh:

   ```
   sqlCopy code
   SELECT department, COUNT(*) as employee_count
   FROM employees
   GROUP BY department;
   ```

   Query ini mengelompokkan data pegawai berdasarkan departemen dan menghitung jumlah pegawai di setiap departemen.

2. **HAVING**:
   - Digunakan bersamaan dengan `GROUP BY` untuk menerapkan kondisi pada hasil agregasi.
   - Ini memfilter hasil agregasi berdasarkan kondisi tertentu.

   Contoh:

   ```
   sqlCopy code
   SELECT department, AVG(salary) as avg_salary
   FROM employees
   GROUP BY department
   HAVING AVG(salary) > 50000;
   ```

   Query ini mengelompokkan pegawai berdasarkan departemen, menghitung rata-rata gaji di setiap departemen, dan hanya mengambil departemen dengan rata-rata gaji di atas 50.000.

3. **MIN**:
   - Mengembalikan nilai minimum dari kolom yang ditentukan.

   Contoh:

   ```
   sqlCopy code
   SELECT MIN(price) as min_price
   FROM products;
   ```

   Ini akan mengambil harga terendah dari semua produk dalam tabel.

4. **MAX**:
   - Mengembalikan nilai maksimum dari kolom yang ditentukan.

   Contoh:

   ```
   sqlCopy code
   SELECT MAX(score) as max_score
   FROM exam_scores;
   ```

   Ini akan mengambil nilai tertinggi dari semua skor ujian.

5. **COUNT**:
   - Mengembalikan jumlah baris dalam hasil query atau jumlah nilai dalam kolom tertentu.

Contoh:

```
sqlCopy code
SELECT COUNT(*) as total_employees
FROM employees;
```

Ini akan menghitung jumlah pegawai dalam tabel.

6. **AVG (AVERAGE)**:
   - Mengembalikan rata-rata nilai dari kolom yang ditentukan.

   Contoh:

   ```
   sqlCopy code
   SELECT AVG(price) as avg_price
   FROM products;
   ```

   Ini akan mengambil rata-rata harga dari semua produk.

7. **SUM**:
   - Mengembalikan total jumlah dari kolom yang ditentukan.

Contoh:

```
sqlCopy code
SELECT SUM(quantity) as total_quantity
FROM order_details;
```

Ini akan menghitung jumlah total kuantitas dari semua detail pesanan.