MODUL PRAKTIKUM X

SELECT III

Tujuan :

- Mampu menampilkan data-data yang ada di tabel menggunakan perintah SQL SELECT
- Mampu memahami klausa SELECT dan kombinasinya

Materi :

- Group By
- Having
- Union
- Aggregate Function
- SubQuery

Persiapan

- Membaca buku literature, referensi atau dari sumber lain tentang penggunaan DML SELECT
- Modul Praktikum X

Landasan Teori

Aggregate Function

Ada lima fungsi aggregate yang bisa digunakan dalam daftar SELECT yang mana kelima ini sudah distandarkan dalam ISO yaitu COUNT, SUM, AVG, MIN, MAX

Aturan – aturan yang berlaku

a. SUM dan AVG digunakan untuk field numeric
b. COUNT, MIN, MAX untuk field numeric dan non numeric
c. COUNT(*) berfungsi menghitung seluruh baris dalam table walaupun terdapat NULL atau duplikasi
d. DISTINCT tidak berpengaruh terhadap operasi MIN atau MAX, tetapi berpengaruh pada SUM atau AVG
e. Fungsi aggregate dapat digunakan dalam SELECT dan HAVING

Subquery

Subquery adalah suatu subselect yang digunakan dalam klausa WHERE dan HAVING disamping SELECT utama. Istilah lainnya adalah Nested Query. Subquery ini juga dapat digunakan dalam perintah INSERT, UPDATE, DELETE

Aturan yang berlaku

1. Klausa ORDER BY dapat tidak digunakan dalam subquery walaupun dapat digunakan dalam SELECT terluar
2. Subquery SELECT harus terdiri atas nama kolom tunggal atau expresi, kecuali untuk subquery yang menggunakan EXISTS
3. Berdasarkan default, nama kolom mengacu ke nama table pada klausa FROM dari subquery, selain itu dapat juga mengacu pada nama alias
4. Karena merupakan sebuah operasi perbandingan, maka harus dituliskan disebelah kanan
5. Tidak dapat digunakan sebagai operasi perbandingan dalam suatu expresi
6. EXISTS dan NOT EXISTS hanya untuk subquery dan menghasilkan hasil benar atau salah
7. EXIST dan NOT EXISTS benar jika dan hanya jika terdapat sedikitnya satu baris dalam table hasil yang dikembalikan oleh subquery, salah jika subquery mengembalikan nilai kosong

Group By

Seringkali informasi yang diinginkan dikelompok berdasarkan kolom-kolom terntu seperti untuk mengetahui frekuensi transaksi supplier dan customer per bulan. Atau untuk mengetahui jumlah siswa yang mengambil kursus, mengetahui berapa siswa yang absen berdasarkan jadwal yang diikuti, mengetahui pembayaran tiap bulan dan lain sebagainya. Untuk pengelompokan data bisa menggunakan klausa GROUP BY dan HAVING sebagai filtering untuk hasil pengelompokan data

Union

Union dipakai untuk menggabungkan dua buah query atau lebih. Syarat du buah query atau lebih bisa digabungkan adalah antara query yang satu dengan yang lainnya harus mempunyai jumlah kolom yang akan ditampilkan harus sama. Demikian juga dengan tipe data, masing – masing kolom yang dimapilkan pada query tersebut, satu sama lainnya harus sama juga

Langkah – Langkah Praktikum

1. Gunakan Database yang telah dibuat pada minggu lalu, yaitu **SBD_XXXX**. XXXX merupakan NIM Masing – masing. Kalau belum ada silakan dibuat terlebih dahulu seperti langkah – langkah pada modul VIII

2. Menampilkan Data menggunakan fungsi Aggregate
    a. Menampilkan rata – rata pembayaran di table tr_payment

```
select avg(total_payment) as [Rata Bayar]
from tr_payment
```

	Rata Bayar
1	1497500.000000

b. Menampilkan no regist_id yang paling besar, dan payment paling besar

```
select max(regist_id), max(total_payment)
from tr_payment
```

	(No column name)	(No column name)
1	reg010	2000000

c. Menampilkan banyaknya total pembayaran yang lebih dari 1500000

```
SELECT COUNT(*) AS banyaknya_pembayaran
FROM tr_payment
WHERE Total_payment > 1500000
```

	banyaknya_pembayaran
1	4

d. Menampilkan jumlah ruangan yang terpakai pada hari senin dan selasa

```
SELECT COUNT(DISTINCT Room) AS Jumlah_ruangan
FROM tr_jadwal
WHERE day_ BETWEEN 'mon' AND 'tue'
```

	Jumlah_ruangan
1	4

e. Menampilkan banyaknya dan total pembayaran

```
SELECT COUNT(total_payment) AS Jumlah_Payment, SUM(total_payment) as Total_Payment
FROM tr_payment
```

	Jumlah_Payment	Total_Payment
1	10	14975000

f. Menampilkan minimal Pembayaran, maximal pembayaran, dan rata2 pembayaran

```
SELECT MIN(total_payment) AS min_payment,
MAX(total_payment) AS max_Payment,
AVG(total_payment) AS avg_payment
FROM tr_payment
```

	min_payment	max_Payment	avg_payment
1	900000	2000000	1497500.000000

3. Menggunakan GROUP BY untuk mengelompokan data

a. Menampilkan jumlah berapa kali masing masing program ID ada jadwalnya

```
SELECT program_id, COUNT(program_id) as myCount
FROM tr_jadwal
GROUP BY program_id
```

	program_id	myCount
1	PC001	2
2	PG001	4
3	PP001	2

b. Menghitung berapa siswa laki dan perempuan

```
SELECT student_gender, COUNT(student_id) as myCount
FROM ms_student
GROUP BY student_gender
```

	student_gender	myCount
1	Female	6
2	male	4

c. Menampilkan berapa kali tutor mengajar

```
SELECT tutor_id, COUNT(tutor_id) as myCount
FROM tr_jadwal
GROUP BY tutor_id
```

	tutor_id	myCount
1	D0612	4
2	D1908	2
3	D1989	2

d. Menampilkan berapa kali tutor id mengajar di ruangan yang sama

```
SELECT tutor_id, room,COUNT(tutor_id) as myCount
FROM tr_jadwal
GROUP BY tutor_id,room
```

	tutor_id	room	myCount
1	D1908	101	2
2	D1989	101	1
3	D0612	102	1
4	D0612	103	1
5	D0612	104	1
6	D1989	104	1
7	D0612	105	1

4. Penggunaan Having

a. Menampilkan program id, jumlah jadwal id yang jadwal id lebih dari satu

```
SELECT program_id, COUNT(jadwal_id) as Banyaknya_Jadwal
FROM tr_jadwal
GROUP BY program_id
HAVING COUNT (jadwal_id) > 1
```

	program_id	Banyaknya_Jadwal
1	PC001	2
2	PG001	4
3	PP001	2

b. Menampilkan jumlah siswa dari hasil written_test dan dikelompokkan berdasarkan program yang diikuti. Yang dimunculkan adalah jumlah siswa yang pesertanya lebih dari 1

Untuk perbandingan, jalankan query dibawah. Query ini untuk menghitung jumlah siswa berdasarkan hasil wrriten_test dan program_id serta ditampilkan dalam bentuk terurut berdasarkan program_id dan written_id

```
select program_id, written_test,count(program_id) as Jml_siswa
from tr_result
group by program_id, written_test
order by program_id, written_test
```

	program_id	written_test	Jml_siswa
1	PC001	A	1
2	PC001	B	1
3	PC001	NULL	1
4	PG001	A	1
5	PG001	C	2
6	PP001	B	2
7	PP001	C	1
8	PP001	NULL	1

Kemudian, dari hasil query diatas, difilter lagi untuk menampilkan jumlah siswa yang lebih besar dari 1

```
select program_id, written_test,count(program_id) as Jml_siswa
from tr_result
group by program_id, written_test
having count(program_id) > 1
order by program_id, written_test
```

	program_id	written_test	Jml_siswa
1	PG001	C	2
2	PP001	B	2

5. Penggunaan Union
   a. Menampilkan seluruh student_id yang telah aktif atau baru mendaftar

```
SELECT student_id
FROM ms_student
WHERE student_id IS NOT NULL
UNION
SELECT student_id
FROM tr_regist
WHERE student_id IS NOT NULL
```

	student_id
1	80692
2	80862
3	82205
4	82312
5	85131
6	81234
7	84321
8	87078
9	83421
10	88139

b. Menampilkan student_id yang hasil ujiannya 'GOOD' atau grade_id 'B'

```
SELECT student_id
FROM tr_result
WHERE note = 'GOOD'
UNION
SELECT student_id
FROM tr_regist
WHERE grade_id = 'B'
```

	student_id
1	80692
2	81234
3	82205
4	82312
5	83421
6	85131
7	88139

c. Menampilkan data siswa dan tutor yang jenis kelaminnya Laki-laki

```
SELECT student_id, student_address, student_gender
FROM ms_student
WHERE student_gender = 'Male'
UNION
SELECT tutor_id, tutor_address, tutor_gender
FROM ms_tutor
WHERE tutor_gender = 'Male'
```

	student_id	student_address	student_gender
1	81234	Slipi	male
2	82205	jembatan tiga	male
3	87078	Jeruk Purut	male
4	88139	gang u	male
5	D1507	serpong	Male
6	D1908	Haji rabu	Male
7	D1989	Harmoni	Male

6. Penggunaan Sub Query

a. Menampilkan jadwal id beserta id tutornya yang programnya PC001

```
SELECT jadwal_id, tutor_id
FROM tr_jadwal
WHERE program_id =
          (SELECT program_id
           FROM ms_program
           WHERE program_id = 'PC001')
```

	jadwal_id	tutor_id
1	J0001	D1908
2	J0007	D1908

b. Menampilkan daftar pembayaran yang pembayarannya melebihi rata – rata pembayaran

```
SELECT payment_id, total_payment, payment_date
FROM tr_payment
WHERE total_payment >
        (SELECT avg(total_payment) FROM tr_payment)
```

	payment_id	total_payment	payment_date
1	PO101	1900000	2008-01-03 00:00:00.000
2	PO104	1900000	2008-02-02 00:00:00.000
3	PO105	2000000	2008-02-05 00:00:00.000
4	PO107	1500000	2008-01-08 00:00:00.000
5	PO109	1500000	2008-01-09 00:00:00.000
6	PO110	1900000	2008-02-12 00:00:00.000

c. Menampilkan student_id, student_name yang absensinya 'hadir' pada pertemuan 5

```
SELECT student_id, student_name
FROM ms_student
WHERE student_id IN(SELECT student_id
        FROM trd_absensi
        WHERE pertemuan_5 = 'hadir')
```

	student_id	student_name
1	80862	Stella clarissa
2	82205	Agustino
3	82312	Imelda putri
4	83421	Juliana
5	84321	Lily Annisa Clarissa
6	85131	Titis annisa astrini
7	87078	Kevin Pratama

d. Menampilkan payment_id yang total paymentnya lebih besar dari total payment dengan regist id reg001 dan reg003 dan reg004

```
SELECT *
FROM tr_payment
WHERE total_payment > all(SELECT total_payment
        FROM tr_payment
WHERE regist_id IN('reg001','reg003', 'reg009'))
```

	payment_id	regist_id	Total_payment	Payment_date
1	PO105	reg005	2000000	2008-02-05 00:00:00.000

e. Menampilkan payment_id yang total paymentnya lebih besar dari total payment dengan regist id reg001 atau reg003 atau reg004

```
SELECT *
FROM tr_payment
WHERE total_payment > any(SELECT total_payment
        FROM tr_payment
WHERE regist_id IN('reg001','reg003', 'reg009'))
```

	payment_id	regist_id	Total_payment	Payment_date
1	PO101	reg001	1900000	2008-01-03 00:00:00.000
2	PO104	reg004	1900000	2008-02-02 00:00:00.000
3	PO105	reg005	2000000	2008-02-05 00:00:00.000
4	PO107	reg007	1500000	2008-01-08 00:00:00.000
5	PO109	reg009	1500000	2008-01-09 00:00:00.000
6	PO110	reg010	1900000	2008-02-12 00:00:00.000