

Pertemuan 9
Database Programming with SQL dan
Syntax SELECT * Statement menggunakan APEX

Tujuan Intruksional :
Pokok Bahasan ini mempelajari tentang *Database Programming* dengan SQL

Kompetensi Yang Diharapkan :
Mahasiswa diharapkan memahami tentang *Database Programming* termasuk cara instalasi dan penulisan *syntax SQL* menggunakan aplikasi *Oracle Application Express (APEX)*.

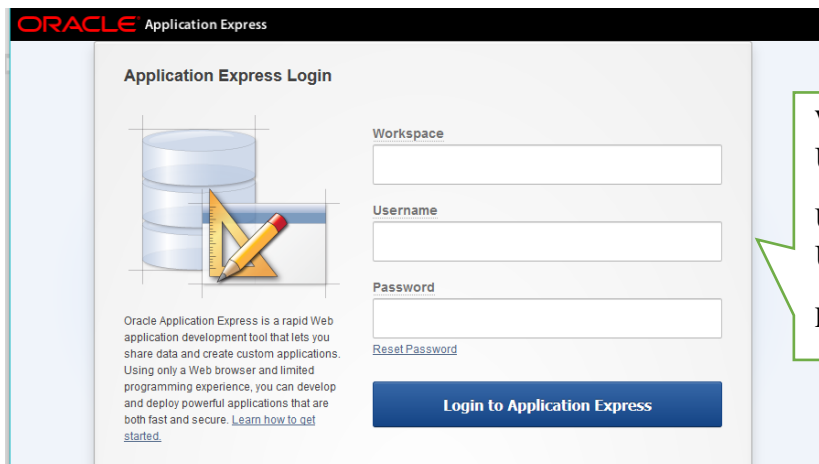
Waktu Pertemuan : 100 Menit

1-1. ORACLE APPLICATION EXPRESS (APEX)

Oracle Application Express (APEX) adalah sebuah *software* yang disediakan oleh Oracle untuk membangun dan mengakses aplikasi seolah-olah bekerja di database terpisah, serta mempercepat proses pengembangan aplikasi. 3 Komponen APEX : *SQL Workshop, Application Builder, Object Browser*.

Log in ke Oracle Application Express (APEX)

Akses halaman *login* APEX : <https://iacademy.oracle.com/>



Workspace :
US_A001APAC6416
Username :
US_A001APAC6416_SQL01_S01
Password : 123456

SQL WORKSHOP → SQL Commands

The screenshot shows the Oracle Application Express SQL Workshop interface. The top navigation bar includes 'Home', 'Application Builder', 'SQL Workshop', and 'Administration'. The 'SQL Workshop' tab is selected, and the 'SQL Commands' sub-tab is active. The 'Schema' dropdown is set to 'US_A001APAC6416_SQL01_S01'. The 'Rows' dropdown is set to '10'. The 'Run' button is highlighted with a red arrow and labeled '1. Run SQL button : Klik tombol ini untuk mengeksekusi pernyataan SQL.'. The 'Statement Window' contains the SQL command 'SELECT * FROM employees;' and is labeled '2. Statement Window: ketik perintah SQL di sini'. The 'Results Window' displays the output of the SQL command as a table with columns: EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, and DEPARTMENT_ID. The table contains 14 rows of data. The 'Results' tab is selected, and the 'Results' window is labeled '3. Result Window : Output dari perintah SQL (atau pesan kesalahan) akan ditampilkan'.

Basic SELECT Statement

1. **SELECT *** : mengambil semua baris pada tabel.

Syntax dasar	Contoh
SELECT * FROM <table name>;	SELECT * FROM employees;

2. **SELECT** Statement dengan sebuah **Condition**

Ubah pernyataan SELECT: mengembalikan subset dari data.

Syntax dasar	Contoh
SELECT <column name 1, column name 2, etc.> FROM <table name> WHERE <condition>;	SELECT first_name, last_name, job_id FROM employees WHERE job_id = 'SA_REP';

Hasil:

FIRST_NAME	LAST_NAME	JOB_ID
Ellen	Abel	SA_REP
Jonathon	Taylor	SA_REP
Kimberely	Grant	SA_REP

3. *Correcting errors*

Saat memasukkan perintah SQL, gunakan ejaan yang benar, jika tidak maka akan mendapatkan pesan *error*.

Contoh:

1. Salah ejaan SELECT

```
SELECT *
FROM employees;
```

ORA-00900: invalid SQL statement

2. Salah memasukkan nama tabel employees .

```
SELECT *
FROM employee;
```



ORA-00942: table or view does not exist

3. Salah memasukkan kolom first_name

```
SELECT name
FROM employees;
```



ORA-00904: "NAME": invalid identifier

1-2. RELATIONAL DATABASE

Relational Database menghubungkan tabel dengan menggunakan *field* yang sama, dengan minimal 2 tabel.

Contoh

Tabel "countries" yang ditampilkan adalah salah satu dari beberapa tabel di database employee.

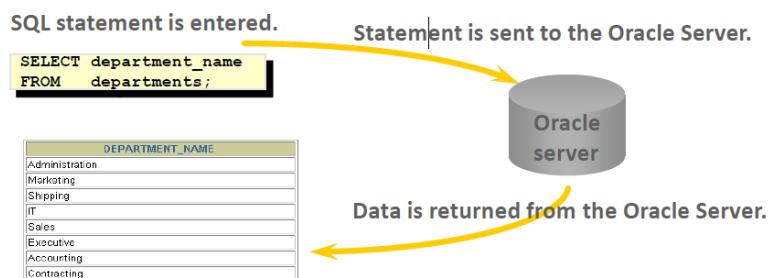
```
SELECT *  
FROM countries;
```

Hasil:

COUNTRY_ID	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

Accessing Data in an RDBMS

Relational Database-Management System (RDBMS) mengatur data ke dalam baris dan kolom yang terkait dengan memasukkan pernyataan SQL di APEX. Permintaan tersebut kemudian dikirim ke Oracle Server (sebuah database yang berjalan di komputer), lalu diproses dan data akan ditampilkan.



Categories of SQL Statements

1. *Data Manipulation Language* (DML): dimulai dengan INSERT, UPDATE, DELETE, atau MERGE yang digunakan untuk memodifikasi data tabel dan memasukkan baris baru, mengubah baris yang ada, atau menghapus baris yang ada.
2. *Data Definition Language* (DDL) : membuat, mengubah, dan menghapus struktur data dari database dimulai dengan CREATE, ALTER, DROP, RENAME, dan TRUNCATE begin DDL statements.
3. *Transaction Control Language* (TCL) : mengelola perubahan yang dibuat oleh pernyataan DML menggunakan COMMIT, ROLLBACK, dan SAVEPOINT, dapat dikelompokkan menjadi logical transactions.
4. *Data Control Language* (DCL): memberi atau menghapus hak akses ke database dan struktur di dalamnya menggunakan GRANT dan REVOKE.

1-3. ANATOMY OF A SQL STATEMENT

SELECT Keyword : mencari data tertentu.

SELECT Statement : mengambil informasi dari database.

Syntax dasar:

```
SELECT <column_name(s)>  
FROM <table_name>;
```

SELECT : menentukan kolom yang akan ditampilkan.

FROM : menentukan tabel yang berisi kolom yang tercantum dalam klausa SELECT.

Konvensi

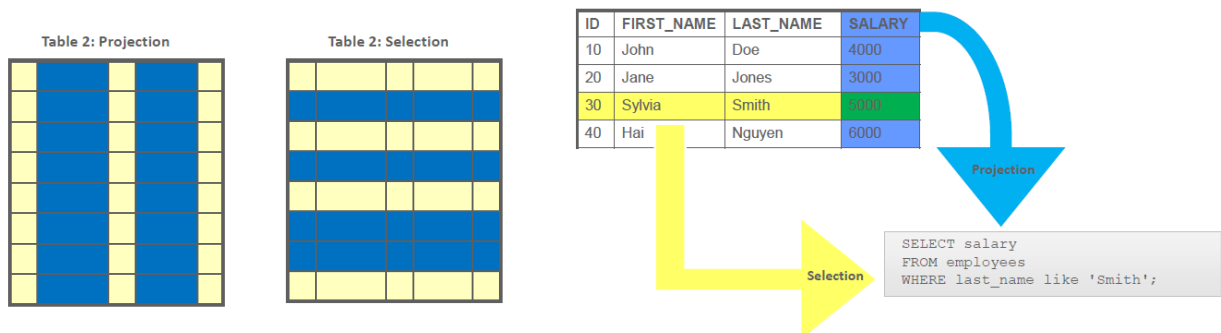
SELECT last_name **FROM** employees

← sebuah klausa (clause): bagian dari pernyataan SQL.
← sebuah pernyataan: kombinasi dari dua / lebih klausa.
← keyword

Projection and Selection

Projection: memilih kolom dalam sebuah tabel

Selection: memilih baris dalam sebuah tabel.



A. Selecting All Columns: menampilkan semua kolom: gunakan *asterisk symbol (*)* dari sebuah nama kolom dalam klausa SELECT.

Contoh:

1. Menampilkan tabel `countries`

```
SELECT *  
FROM countries;
```

2. Menampilkan semua kolom dalam tabel menggunakan daftar masing-masing.

```
SELECT country_id, country_name, region_id  
FROM countries;
```

Hasil:

COUNTRY_ID	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

B. Projecting Specific Columns: menampilkan PROJECT dengan kolom tertentu dari tabel. Masukkan nama kolom dan pisahkan setiap nama dengan koma (.).

```
SELECT location_id, city, state_province  
FROM locations;
```

Hasil:

LOCATION_ID	CITY	STATE_PROVINCE
1800	Toronto	Ontario
2500	Oxford	Oxford
1400	Southlake	Texas
1500	South San Francisco	California
1700	Seattle	Washington

Menggunakan Operator Aritmatika

- add (+), subtract (-) , multiply (*) and divide (/)
- Hasil perhitungan hanya ditampilkan pada *output*, tidak membuat sebuah kolom baru pada tabel/ mengganti nilai data.
- Contoh: Kolom nilai awal `SALARY` dan *output* `SALARY + 300` untuk semua `employees`.

```
SELECT last_name, salary, salary + 300
FROM employees;
```

Hasil:

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Whalen	4400	4700
Higgins	12000	12300
Gietz	8300	8600
Zlotkey	10500	10800
Abel	11000	11300
Taylor	8600	8900
Grant	7000	7300

Precedence in Arithmetic Operators

- Precedence adalah urutan di mana Oracle mengevaluasi operator yang berbeda dalam ekspresi yang sama.
- Saat mengevaluasi sebuah ekspresi yang mengandung banyak operator, Oracle mengevaluasi operator dengan prioritas yang lebih tinggi sebelum mengevaluasi yang lebih rendah.
- Oracle mengevaluasi operator dengan prioritas yang sama dari kiri ke kanan dalam sebuah ekspresi.
- Gunakan tanda kurung () untuk mendahulukan ekspresi yang dievaluasi.

Operator Precedence (Tidak menggunakan tanda kurung)

```
SELECT last_name, salary, 12*salary +100
FROM employees;
```

Hasil:

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100
Whalen	4400	52900
Higgins	12000	144100
Gietz	8300	99700
Zlotkey	10500	126100
Abel	11000	132100
Taylor	8600	103300
Grant	7000	84100

Using Parentheses (Menggunakan tanda kurung)

```
SELECT last_name, salary, 12*(salary +100)
FROM employees;
```

Hasil:

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Whalen	4400	54000
Higgins	12000	145200
Gietz	8300	100800
Zlotkey	10500	127200
Abel	11000	133200
Taylor	8600	104400
Grant	7000	85200

NULL Values

- NULL adalah nilai yang unavailable, unassigned, unknown, atau inapplicable.
- NULL tidak sama dengan nol atau spasi.
- Relational databases menggunakan *placeholder* yang disebut NULL / null untuk mewakili nilai-nilai yang tidak diketahui.

Contoh:

```
SELECT last_name, job_id, salary, commission_pct, salary*commission_pct
FROM employees;
```

Hasil:

Salaries dan Commissions

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT	SALARY*COMMISSION_PCT
King	AD_PRES	24000	-	-
Kochhar	AD_VP	17000	-	-
De Haan	AD_VP	17000	-	-
Whalen	AD_ASST	4400	-	-
Higgins	AC_MGR	12000	-	-
Gietz	AC_ACCOUNT	8300	-	-
Zlotkey	SA_MAN	10500	.2	2100
Abel	SA_REP	11000	.3	3300
Taylor	SA_REP	8600	.2	1720
Grant	SA_REP	7000	.15	1050

Aliases (AS)

- Alias adalah penamaan kembali sebuah judul kolom pada *output*.
- Tanpa alias, ketika hasil pernyataan SQL ditampilkan, nama kolom yang ditampilkan akan sama dengan nama kolom pada tabel atau nama yang menunjukkan operasi aritmatika seperti $12 * (SALARY + 100)$.
- Kolom alias: Mengganti nama judul, berguna dalam perhitungan, diikuti nama kolomnya, bisa terdapat keyword AS opsional antara nama kolom dan alias, menggunakan tanda kutip (“ ”) jika berisi spasi / karakter khusus / *case-sensitive*.

Menggunakan Kolom Aliases

- *Syntax* dasar:

```
SELECT * |column|expr [ AS alias], .....  
FROM table;
```

- Contoh:

```
SELECT last_name AS name,  
       commission_pct AS comm  
FROM employees;
```

NAME	COMM
King	-
Kochhar	-
De Haan	-

```
SELECT last_name "Name", salary*12  
       "Annual Salary"  
FROM employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

2-1 COLUMNS, CHARACTERS, AND ROWS

DESCRIBE

DESCRIBE (DESC) digunakan untuk menampilkan struktur tabel (nama tabel, tipe data, *Primary Key* dan *Foreign Key*, *Nullable*).

Sintaks dasar	Contoh
DESC <nama_tabel>;	DESC customers;

Hasil :

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMERS	CUSTID	VARCHAR2	5	-	-	1	-	-	-
	COMPANYNAME	VARCHAR2	30	-	-	-	✓	-	-
	CONTACTNAME	VARCHAR2	30	-	-	-	✓	-	-
	ADDRESS	VARCHAR2	30	-	-	-	✓	-	-
	CITY	VARCHAR2	20	-	-	-	✓	-	-
	STATE	VARCHAR2	30	-	-	-	✓	-	-
	PHONE	VARCHAR2	20	-	-	-	✓	-	-
1 - 7									

Operator Concatenation (Penggabungan)

Concatenation (Penggabungan) berfungsi untuk menggabungkan dua atau lebih karakter menjadi satu kesatuan kalimat. Simbol concatenation adalah 2 garis vertikal || atau disebut “pipes”.

Sintaks dasar :

```
string1 || string2 || string_n
```

Contoh :

- Concatenation

```
SELECT department_id || department_name  
FROM departments;
```

Hasil :

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
50	Shipping
60	IT
80	Sales
90	Executive
110	Accounting
190	Contracting

- Concatenation dengan Literal Values

```
SELECT last_name || ' has a monthly salary of ' ||
       salary || ' dollars.' AS Pay
FROM employees;
```

Hasil :

PAY
King has a monthly salary of 24000 dollars.
Kochhar has a monthly salary of 17000 dollars.
De Haan has a monthly salary of 17000 dollars.
Whalen has a monthly salary of 4400 dollars.
Higgins has a monthly salary of 12000 dollars.

- Concatenation dengan Alias

```
SELECT first_name || ' ' || last_name AS "Employee Name"
FROM employees;
```

Hasil :

Employee Name
Ellen Abel
Curtis Davies
Lex De Haan
Bruce Ernst
Pat Fay

2-2 LIMIT ROWS SELECTED

DISTINCT

DISTINCT digunakan adalah salah satu operator di database Oracle bahkan hampir di semua database yang digunakan untuk mencegah adanya duplikasi data atau record. Misalkan ada 10 orang yang bernama 'AHMAD', maka dengan menggunakan operator Distinct, bisa mengeliminasi data 'AHMAD' menjadi satu nama saja. DISTINCT digunakan setelah klausa SELECT.

Sintaks dasar	Contoh
SELECT DISTINCT <nama_kolom> FROM <nama_tabel>;	SELECT DISTINCT department_id FROM employees;

Hasil :

DEPARTMENT_ID
-
90
20
110
80
50
10
60

WHERE

Klausa WHERE pada perintah SELECT digunakan untuk menyeleksi data atau record sesuai dengan kondisi yang diinginkan. Klausa WHERE bersifat opsional.

a. WHERE dengan Operator Perbandingan

Contoh :

```
SELECT employee_id, last_name, department_id  
FROM employees  
WHERE department_id = 90;
```

Hasil :

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90

b. WHERE dengan Karakter

Contoh :

```
SELECT first_name, last_name  
FROM employees  
WHERE last_name = 'Taylor';
```

Hasil :

FIRST_NAME	LAST_NAME
Jonathon	Taylor

2-3 COMPARISON OPERATORS (OPERATOR PERBANDINGAN)

BETWEEN...AND

Operator ini digunakan untuk memilih dan menampilkan baris berdasarkan range nilai.

Range nilai yang ditampilkan termasuk nilai batas terendah dan nilai batas tertinggi.

Contoh :

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 9000 AND 11000;
```

Hasil :

LAST_NAME	SALARY
Zlotkey	10500
Abel	11000
Hunold	9000

IN

IN digunakan untuk menguji apakah sebuah nilai berada dalam seperangkat nilai tertentu.

Contoh :

```
SELECT city, state_province, country_id
FROM locations
WHERE country_id IN('UK', 'CA');
```

Hasil :

CITY	STATE_PROVINCE	COUNTRY_ID
Toronto	Ontario	CA
Oxford	Oxford	UK

LIKE

Perintah LIKE merupakan kondisi untuk mendapatkan data dengan memilih data yang sesuai dengan kondisi. Dalam melakukan pencarian dengan kondisi like, maka perlu menyebutkan wildcard berupa garis bawah (_) atau persen (%).

Berikut penjelasannya :

- Tanda garis bawah (_) berarti cocok dengan sebuah karakter apa saja dengan panjang karakter harus sesuai dengan jumlah karakter garis bawah. Contohnya like a_i berarti cocok dengan ani, adi atau ali namun tidak cocok dengan abri atau andi karena garis bawahnya hanya satu karakter diantara a dan i.
- Tanda persen (%) berarti cocok dengan karakter apa saja tanpa bergantung panjangnya.

Contoh :

Mencari last_name pada tabel employees yang memiliki huruf kedua terakhir 'o'
last_name.

```
SELECT last_name
FROM employees
WHERE last_name LIKE '_o%';
```

Hasil :

LAST_NAME
Kochhar
Lorentz
Mourgos

Opsi ESCAPE dapat digunakan untuk menunjukkan bahwa _ atau % adalah bagian dari namanya, bukan nilai wildcard.

Contoh :

Mencari JOB_ID dari tabel employees yang berisi pola _R.

```
SELECT last_name, job_id
FROM EMPLOYEES
WHERE job_id LIKE '%\_R%' ESCAPE '\';
```

Hasil :

LAST_NAME	JOB_ID
Abel	SA_REP
Fay	MK_REP
Grant	SA_REP
Taylor	SA_REP

IS NULL, IS NOT NULL

Tes kondisi IS NULL digunakan untuk data yang tidak tersedia, belum ditetapkan, atau tidak diketahui. IS NOT NULL digunakan untuk data yang tersedia di database.

Contoh :

- Mencari semua last_name dari employees yang tidak memiliki manajer.

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL;
```

Hasil :

LAST_NAME	MANAGER_ID
King	-

- IS NOT NULL menampilkan baris yang memiliki nilai di kolom commission_pct.

```
SELECT last_name, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL;
```

Hasil :

LAST_NAME	COMMISSION_PCT
Zlotkey	.2
Abel	.3
Taylor	.2
Grant	.15

3-1 LOGICAL COMPARISONS AND PRECEDENCE RULES

Logical Operators: menggabungkan dua atau lebih kondisi untuk menghasilkan satu hasil.

Returned ONLY IF keseluruhan hasil dari kondisi ini benar.

AND : Returns TRUE jika kedua kondisi benar.

OR : Returns TRUE jika salah satu kondisi benar.

NOT : Returns TRUE jika kondisi salah.

Operator AND

Hasil yang dikembalikan akan menjadi baris yang memenuhi KEDUA kondisi yang ditentukan dalam klausa WHERE.

```
SELECT last_name, department_id, salary
FROM employees
WHERE department_id > 50 AND salary > 12000;
```

LAST_NAME	DEPARTMENT_ID	SALARY
King	90	24000
Kochhar	90	17000
De Haan	90	17000

```
SELECT last_name, hire_date, job_id
FROM employees
WHERE hire_date > '01/jan/1998' AND job_id
      LIKE 'SA%';
```

LAST_NAME	HIRE_DATE	JOB_ID
Zlotkey	29-Jan-2000	SA_MAN
Taylor	24-Mar-1998	SA_REP
Grant	24-May-1999	SA_REP

Operator OR

- Jika klausa WHERE menggunakan kondisi OR, hasil yang dikembalikan dari kueri akan menjadi baris yang memenuhi salah satu dari kondisi OR.
- Semua baris kembali memiliki location_id = 2500 OR mereka memiliki seorang manager_id = 124.

```
SELECT department_name, manager_id, location_id
FROM departments
WHERE location_id = 2500 OR manager_id=124;
```

Hasil:

DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
Shipping	124	1500
Sales	149	2500

Operator NOT

Mengembalikan baris yang TIDAK memenuhi syarat dalam klausa WHERE.

```
SELECT department_name, location_id
FROM departments
WHERE location_id NOT IN (1700,1800);
```

Hasil:

DEPARTMENT_NAME	LOCATION_ID
Shipping	1500
IT	1400
Sales	2500

Aturan yang diutamakan

AND-OR

```
SELECT last_name||' '||salary*1.05
AS "Employee Raise"
FROM employees
WHERE department_id IN(50,80)
AND first_name LIKE 'C%'
OR last_name LIKE '%s%';
```

1. Operator AND dievaluasi sebelum operator OR.
2. Jika salah satu dari kondisi dalam pernyataan AND tidak terpenuhi, maka operator OR digunakan untuk memilih baris.

Hasil:

Employee Raise
Higgins 12600
Mourgos 6090
Rajs 3675
Davies 3255
Matos 2730
Vargas 2625
Ernst 6300
Hartstein 13650

- Konsep penting:

ORDER	OPERATORS
1	Aritmatika + - * /
2	Concatenation
3	Comparison <, <=, >, >=, <>
4	IS (NOT) NULL, LIKE, (NOT) IN
5	(NOT) BETWEEN
6	NOT
7	AND
8	OR

AND – OR

Contoh:

```
SELECT last_name||' '||salary*1.05
AS "Employee Raise",department_id, first_name
FROM employees
WHERE department_id IN(50,80)
AND first_name LIKE 'C%'
OR last_name LIKE '%s%';
```

1. Kondisi AND dievaluasi, sehingga semua karyawan yang bekerja di dept 80 atau 50, AND yang memiliki nama depan dimulai dengan huruf "C" dikembalikan.
2. Klausa OR kemudian dievaluasi dan mengembalikan employees dengan nama terakhir huruf "s".

Hasil:

Employee Raise	DEPARTMENT_ID	FIRST_NAME
Higgins 12600	110	Shelley
Mourgos 6090	50	Kevin
Rajs 3675	50	Trenna
Davies 3255	50	Curtis
Matos 2730	50	Randall
Vargas 2625	50	Peter
Ernst 6300	60	Bruce
Hartstein 13650	20	Michael

OR-AND

Contoh:

```
SELECT last_name||' '||salary*1.05
AS "Employee Raise", department_id, first_name
```

```

FROM employees
WHERE department_id IN(50,80)
OR first_name LIKE 'C%'
AND last_name LIKE '%s%';

```

1. first_name dimulai dengan "C" AND last_name berisi "s". Kedua kondisi ini harus dipenuhi untuk dikembalikan.
2. Setiap employees di department 50 dan 80 akan dikembalikan.

Hasil:

Employee Raise	DEPARTMENT_ID	FIRST_NAME
Zlotkey 11025	80	Eleni
Abel 11550	80	Ellen
Taylor 9030	80	Jonathon
Mourgos 6090	50	Kevin
Rajs 3675	50	Trenna
Davies 3255	50	Curtis
Matos 2730	50	Randall
Vargas 2625	50	Peter

Menambahkan tanda kurung akan mengubah cara klausa WHERE dievaluasi, dan baris kembali.

```

SELECT last_name||' '||salary*1.05
AS "Employee Raise", department_id, first_name
FROM employees
WHERE (department_id IN(50,80)
OR first_name LIKE 'C%')
AND last_name LIKE '%s%';

```

1. Nilai dalam tanda kurung dipilih.
2. Semua nilai dalam tanda kurung yang juga mengandung huruf "s" pada last_name akan dikembalikan.

Hasil:

Employee Raise	DEPARTMENT_ID	FIRST_NAME
Mourgos 6090	50	Kevin
Rajs 3675	50	Trenna
Davies 3255	50	Curtis
Matos 2730	50	Randall
Vargas 2625	50	Peter

3-2 SORTING ROWS

ORDER BY

ORDER BY digunakan untuk mengurutkan data.

Contoh :

- **Ascending**

```
SELECT last_name, hire_date
FROM employees
ORDER BY hire_date;
```

Hasil :

LAST_NAME	HIRE_DATE
King	17-Jun-1987
Whalen	17-Sep-1987
Kochhar	21-Sep-1989
Hunold	03-Jan-1990
Ernst	21-May-1991

- **Descending**

```
SELECT last_name, hire_date
FROM employees
ORDER BY hire_date DESC;
```

Hasil :

LAST_NAME	HIRE_DATE
Zlotkey	29-Jan-2000
Mourgos	16-Nov-1999
Grant	24-May-1999
Lorentz	07-Feb-1999
Vargas	09-Jul-1998

- **Menggunakan kolom Alias**

```
SELECT last_name, hire_date AS "Date Started"
FROM employees
ORDER BY "Date Started";
```

Hasil :

LAST_NAME	Date Started
King	17-Jun-1987
Whalen	17-Sep-1987
Kochhar	21-Sep-1989
Hunold	03-Jan-1990
Ernst	21-May-1991

- **Menggunakan kolom lain**

```
SELECT employee_id, first_name
FROM employees
WHERE employee_id < 105
ORDER BY last_name;
```

Hasil :

EMPLOYEE_ID	FIRST_NAME
102	Lex
104	Bruce
103	Alexander
100	Steven
101	Neena

- **Menggunakan banyak kolom**

```
SELECT department_id, last_name
FROM employees
WHERE department_id <= 50
ORDER BY department_id, last_name;
```

Hasil :

DEPARTMENT_ID	LAST_NAME
10	Whalen
20	Fay
20	Hartstein
50	Davies
50	Matos
50	Mourgos
50	Rajs
50	Vargas

TUGAS MINGGUAN

Tuliskan *syntax* untuk menghasilkan *output* sebagai berikut:

- Petunjuk:** 1. Semua soal menggunakan tabel `employees`.
2 No. soal sesuai nomor materi.

1-1

FIRST_NAME	LAST_NAME	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
Trenna	Rajs	17-Oct-1995	ST_CLERK	3500	50
Curtis	Davies	29-Jan-1997	ST_CLERK	3100	50
Randall	Matos	15-Mar-1998	ST_CLERK	2600	50
Peter	Vargas	09-Jul-1998	ST_CLERK	2500	50

1-3

FIRST_NAME	JOB_ID	SALARY	Annual Salary
Steven	AD_PRES	24000	48200
Neena	AD_VP	17000	34200
Lex	AD_VP	17000	34200
Jennifer	AD_ASST	4400	9000
Shelley	AC_MGR	12000	24200
William	AC_ACCOUNT	8300	16800
Eleni	SA_MAN	10500	21200
Ellen	SA_REP	11000	22200
Jonathon	SA_REP	8600	17400
Kimberely	SA_REP	7000	14200

2-1

EMAIL	JOB_ID	PAY
SKING	AD_PRES	King has a monthly salary of 240000 dollars.
NKOCHHAR	AD_VP	Kochhar has a monthly salary of 170000 dollars.
LDEHAAN	AD_VP	De Haan has a monthly salary of 170000 dollars.
JWHALEN	AD_ASST	Whalen has a monthly salary of 44000 dollars.
SHIGGINS	AC_MGR	Higgins has a monthly salary of 120000 dollars.
WGIEZT	AC_ACCOUNT	Gietz has a monthly salary of 83000 dollars.
EZLOTKEY	SA_MAN	Zlotkey has a monthly salary of 105000 dollars.
EABEL	SA_REP	Abel has a monthly salary of 110000 dollars.
JTAYLOR	SA_REP	Taylor has a monthly salary of 86000 dollars.
KGRANT	SA_REP	Grant has a monthly salary of 70000 dollars.

2-2

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
124	Mourgos	50
141	Rajs	50
142	Davies	50
143	Matos	50
144	Vargas	50

2-3

FIRST_NAME	SALARY
William	8300
Jonathon	8600
Kimberely	7000
Kevin	5800
Alexander	9000
Bruce	6000
Pat	6000

3-1

Employee Raise	HIRE_DATE	MANAGER_ID	DEPARTMENT_ID	FIRST_NAME
Ellen 110000	11-May-1996	149	80	Ellen
Jonathon 86000	24-Mar-1998	149	80	Jonathon
Kimberely 70000	24-May-1999	149	-	Kimberely
Trenna 35000	17-Oct-1995	124	50	Trenna
Curtis 31000	29-Jan-1997	124	50	Curtis
Randall 26000	15-Mar-1998	124	50	Randall
Peter 25000	09-Jul-1998	124	50	Peter

3-1

```
SELECT *
FROM employees
WHERE salary <= 10000 AND job_id LIKE 'ST%' OR last_name
LIKE 'a%';
```

Output dari syntax diatas adalah....

3-2

LAST_NAME	Date Started	SALARY	DEPARTMENT_ID	EMAIL
Mourgos	16-Nov-1999	5800	50	KMOURGOS
Grant	24-May-1999	7000	-	KGRANT
Lorentz	07-Feb-1999	4200	60	DLORENTZ
Vargas	09-Jul-1998	2500	50	PVARGAS
Matos	15-Mar-1998	2600	50	RMATOS
Fay	17-Aug-1997	6000	20	PFAY
Davies	29-Jan-1997	3100	50	CDAVIES
Rajs	17-Oct-1995	3500	50	TRAJS
Ernst	21-May-1991	6000	60	BERNST
Whalen	17-Sep-1987	4400	10	JWHALEN