HANDOUT BASIS DATA LANJUT

Genap 2009/2010

Fakultas Teknik
Program Studi SISTEM INFORMASI (S1)
Universitas Kristen Duta Wacana
(U.K.D.W.)

Disusun Oleh:
Yetli Oslan, S.Kom., M.T.

Universitas Kristen "Duta Wacana"
Jl. Dr. Wahidin S. No. 5 - 25
Yogyakarta 55224

FAKULTAS TEKNIK
PRODI SISTEM INFORMASI
Alamat: Jl. Dr. Wahidin 5 - 25 Yogyakarta 55224
Gedung AGAPE lt.3, telp: 0274 - 563929 ext 303

KONTRAK PERKULIAHAN

Program Studi	:	S1 Sistem Informasi
Mata kuliah	:	Basis Data Lanjut
Kode	:	TP4013
Sks / Harga	:	3 / 4 sks (aktivitas di Lab)
Semester	:	Pilihan Bebas
Mata kuliah Prasyarat	:	Total Sistem Basis Data ≥ D
Dosen	:	Yetli Oslan, S.Kom., MT.
Bentuk Perkuliahan	:	Tatap muka di lab 1x seminggu dengan dosen.

I. Deskripsi Mata Kuliah

Mata kuliah ini merupakan kelanjutan dari mata kuliah Basis Data. Tujuan utama mata kuliah ini adalah memperkenalkan secara langsung kepada mahasiswa tentang lingkungan *database client server* serta berbagai pengetahuan untuk mengelola basis data server dengan menggunakan perintah-perintah SQL menggunakan tools Oracle. Kuliah akan diawali dengan penjelasan tentang basis data yang akan digunakan sebagai studi kasus dilanjutkan dengan membangun, mengelola dan memanipulasinya dengan menggunakan perintah-perintah DML dan DDL. Mahasiswa juga diperkenalkan pada obyek-obyek dalam basis data sampai dengan pengendalian hak akses pada *database client server*.

II. Kompetensi Mata Kuliah

1. Memahami konsep *database client server*
2. Mamahami perintah-perintah DML dan DDL
3. Mampu menciptakan dan mengendalikan obyek-obyek basis data *(table, view, sequence, index, synonym)*
4. Mampu mengendalikan dan mengantur batasan kewenangan setiap user dalam berhubungan dengan basis data

III. Strategi Perkuliahan

A. Tatap muka:
   - ❑ Kuliah tatap muka langsung di LAB
   - ❑ Lain-lain: -

B. Non tatap muka:
   - ❑ Tugas mandiri
   - ❑ Lain-lain: Praktikum

IV. Sumber Bahan

Judul Buku	Pengarang	Penerbit	Tahun
Oracle: A Beginner Guide	Michael Abbey	McGraw Hill	1995
A Guide to Oracle	Joline Morrison	Course Technology - ITP	1998
Oracle 9 Howto	Edward Honour	Waite Group Press	1998
Oracle Programming with Visual Basic	Nick Snowdon	Sybex	1999
Practical File System Design with the be file system		Morgan Kaufmann Publishers	1999
Oracle9I: The Complete Reference	Kevin Loney George Koch	McGraw Hill	2002

V. Penilaian

No.	Jenis Penilaian	Bobot (%)
1.	Kehadiran dan partisipasi kuliah	-
2.	Presentasi & diskusi	-
3.	Tugas-tugas + Tes Mingguan	30
4.	Tes kecil	30
5.	Tes Tengah Semester	20
6.	Tes Akhir Semester	20
7.	Lain-lain: Hadir minimal 75 % untuk dapat ikut Tes Akhir Semester	
Jumlah		100

VI. Kegiatan Perkuliahan

No.	Materi	Keterangan
1.	PENDAHULUAN	Abstraksi Kuliah Basis Data Lanjut (materi + teknis kuliah & praktikum) Mengenal Oracle: Review PK IV dan Basis Data (DBMS, Logical Data Models ER + Type Data + Constraint + Index)
2.	PEMAHAMAN STUDI KASUS	Memahami Basis Data yang akan digunakan dalam praktek (LAB)
3.	DML	Writing Basic SQL Statements Resticting and Sorting Data Single Row Functions Displaying Data from Multiple Tables (Joins) Aggregating Data Using Group Functions Subqueries Manipulating Data: INSERT, DELETE, UPDATE
4.	Transaction Control	Commit, Rollback, Savepoint
5.	DDL	Creating dan Managing ***Table***s Including Constraints Creating ***View***s Others Database Objects: ***Sequence, Index, Synonym***
6.	PL/SQL	Block types, Nested Blocks, SQL Cursor
7.	Controlling User Access (Data Control Language – DCL)	Mengendalikan dan mengatur batasan kewenangan setiap user dalam berhubungan dengan data
8.	Advanced	Hierarchical Retrieval Oracle *9i* Datetime Functions

E-MAIL: yetli@ukdw.ac.id

Disepakati untuk diberlakukan pada:
Genap 2009/2010, Grup: A

Dosen Pengampu:

Yetli Oslan, S.Kom., M.T.

1

Oracle 9*i* → *10g* → *11g*

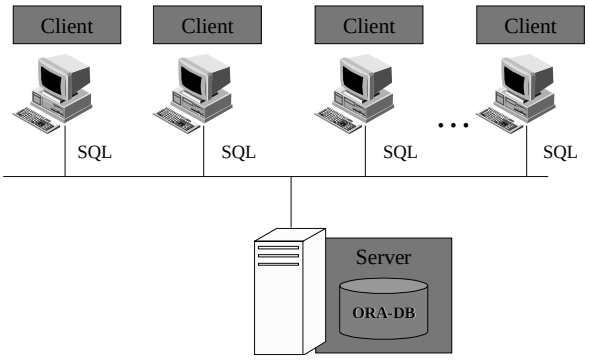
- Oracle 9*i* Application Server (Oracle9*i*AS)
  Untuk menjalankan semua aplikasi.
- Oracle 9*i* Database
  - RDBMS
  - ORDBMS (mulai Oracle 8)
  - Spreadsheets
  - Word documents
  - Powerpoint presentations
  - XML
  - Multimedia : MP3, Grafik, Video dll

2

Oracle 9*i*

- Oracle 9*i* Database
  - Struktur: Client – Server

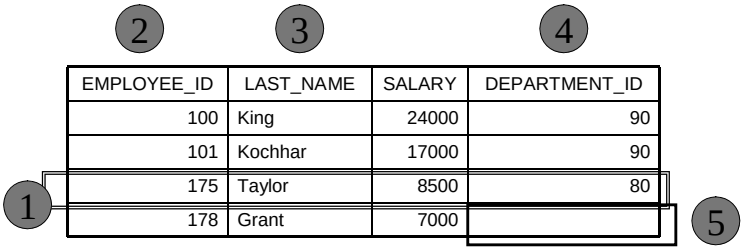
    Client menggunakan interface seperti ODBC *(Open Database Conectivity)* dan JDBC *(Java Database Conectivity)* untuk dapat berhubungan dengan server



Struktur umum sistem Client - Server

4

Relational Database Terminology



EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
100	King	24000	90
101	Kochhar	17000	90
175	Taylor	8500	80
178	Grant	7000	

1. Sebuah baris yang menyajikan semua informasi yang diperlukan dari seorang pegawai
2. Kolom tentang employee_id. Berisi nomor yang mengidentifikasi setiap pegawai secara unik. Dalam tabel, kolom ini diset sebagai Primary Key: harus mengandung sebuah nilai (tdk boleh kosong) dan harus unik.
3. Sebuah kolom yang bukan kunci, sebagai bagian dari informasi setiap pegawai.
4. Kolom tentang department_id. Juga merupakan Foreign Key: setiap nilai yang tercatat dikolom ini, harus punya relasi ke tabel induknya (tabel DEPARTMENTS)
5. Sebuah field mungkin tidak diisi nilai, ini disebut NULL

5

Primary Key & Foreign Key…(1)

- Kita tidak bisa memasukkan nilai yang sama (duplikat) dalam sebuah *PK*
- Secara umum *PK* tidak boleh diubah
- *FK* cenderung berperan sebagai pointer
- Nilai pada *FK* harus *match* dengan nilai pada *PK*
- Sebuah *FK* harus mempunyai referensi pada sebuah kolom *PK*

6

Primary Key & Foreign Key…(2)

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
100	King	24000	90
101	Kochhar	17000	90
175	Taylor	8500	80
178	Grant	7000	

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1600
50	Shipping	124	1500
60	IT	100	1400
80	Sales	149	2500
90	Executive	100	1700

7

Stuctured Query Language (SQL)

- Dengan menggunakan SQL kita dapat berkomunikasi dengan Oracle Database Server.

Keuntungan:

- Efisien
- Mudah dipelajari dan digunakan
- Mempunyai fungsi yang lengkap *(define, retrieve, manipulate data in the tables)*

8

SQL Statement

SELECT	Data retrieval
INSERT UPDATE DELETE MERGE	Data Manipulation Language (DML)
CREATE ALTER DROP RENAME TRUNCATE	Data Definition Language (DDL)
COMMIT ROLLBACK SAVEPOINT	Transaction Control (untuk DML)
GRANT REVOKE	Data Control Language (DCL)

9

Character Functions

Function	Sample
LOWER*(column/expression)*	LOWER('SQL Course')
UPPER*(column/expression)*	UPPER('SQL Course')
INITCAP*(column/expression)*	INITCAP('SQL Course')
CONCAT*(column1/expression1 , column2/expression2)*	CONCAT('Hello', 'World')
SUBSTR*(column/expression, m [,n])*	SUBSTR('HelloWorld',1,5)
LENGTH*(column/expression)*	LENGTH('HelloWorld')
INSTR*(column/expression, 'string', [m] [,n])*	INSTR('HelloWorld', 'W')
LPAD*(column/expression, n, 'string')* RPAD*(column/expression, n, 'string')*	LPAD(salary,10,'*') RPAD(salary,10,'*')
TRIM*(leading/trailing/both, trim_character FROM trim_source)*	TRIM('H' FROM 'HelloWorld')
REPLACE*(text, search_string, replacement_string)*	REPLACE('Jadwal','w','u')

Number Functions

10

```
SELECT   ROUND(45.923,2), ROUND(45.923,0), ROUND(45.923,-1)
FROM     DUAL;
```

```
SELECT   TRUNC(45.923,2), TRUNC(45.923), TRUNC(45.923,-2)
FROM     DUAL;
```

```
SELECT   last_name, salary, MOD(salary,5000)
FROM     employees
WHERE    job_id = 'SA_REP';
```

DUAL adalah tabel dummy yang dapat digunakan untuk melihat hasil dari suatu fungsi dan kalkulasi

Date Functions

11

- Oracle menyimpan date dalam format: century, year, month, day, hours, minutes, second
- Default display: DD-MON-RR
- SYSDATE adalah fungsi yang akan mengembalikan informasi tentang date & time dari system

Operation	Result
date + number	Date (menambah hari dari suatu tanggal)
date – number	Date (mengurangi hari dari suatu tanggal)
date-date	Number of days (mengurangi tanggal dengan tanggal untuk menghitung hari)
date + number/24	Date (menambah jam ke tanggal)

```
SELECT   last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM     employees
WHERE    department_id = 90;
```

Date Functions

12

Function	Description
MONTHS_BETWEEN	Number of months between two dates MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')
ADD_MONTHS	Add calendar months to date ADD_MONTHS ('11-JAN-94',6)
NEXT_DAY	Next day of the date specified NEXT_DAY('01-SEP-95', 'FRIDAY')
LAST_DAY	Last day of the month LAST_DAY('01-SEP-95')
ROUND	Round date ROUND(SYSDATE,'MONTH') ROUND(SYSDATE,'YEAR')
TRUNC	Truncate date TRUNC(SYSDATE,'MONTH') TRUNC(SYSDATE,'YEAR')

Displaying Data from Multiple Tables…(1)

13

- Using Table Aliases

```
SELECT   e.last_name, d.departement_name, l.city
FROM     employees e, departments d, locations l
WHERE    e. department_id = d.department_id
         AND d.location_id = l.location_id;
```

- Joins

```
SELECT   e.last_name, e.department_id, d.departement_name
FROM     employees e, departments d
WHERE    e. department_id(+) = d.department_id;
```

OUTER JOINS

Outer joins simbol

```
SELECT   worker.last_name || ' works for '|| manager.last_name
FROM     employees worker, employees manager
WHERE    worker.manager_id = manager.employee_id;
```

SELF JOINS

14

Displaying Data from Multiple Tables…(2)

- Joins

CROSS JOINS

```
SELECT   last_name, department_name
FROM     employees
CROSS JOIN departments;
```

NATURAL JOINS

```
SELECT   department_id, department_name, location_id, city
FROM     departments
NATURAL JOIN locations;
```

LEFT OUTER JOIN

```
SELECT   e.last_name, e.department_id, d.department_name
FROM     employees e
LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

RIGHT OUTER JOIN

```
SELECT   e.last_name, e.department_id, d.department_name
FROM     employees e
RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

FULL OUTER JOIN

```
SELECT   e.last_name, e.department_id, d.department_name
FROM     employees e
FULL OUTER JOIN departments d
ON (e.department_id = d.department_id;
```

15

GROUP Functions

- AVG --- AVG([DISTINCT|ALL]n)
- COUNT --- COUNT({*|[DISTINCT|ALL]expr})
- MAX --- MAX([DISTINCT|ALL]expr)
- MIN --- MIN([DISTINCT|ALL]expr)
- STDDEV --- STDDEV([DISTINCT|ALL]x)
- SUM --- SUM([DISTINCT|ALL]n)
- VARIANCE --- VARIANCE([DISTINCT|ALL]x)

```
SELECT      [column,] group_function(column), …
FROM        table
[WHERE      condition]
[GROUP BY   column]
[ORDER BY   column];
```

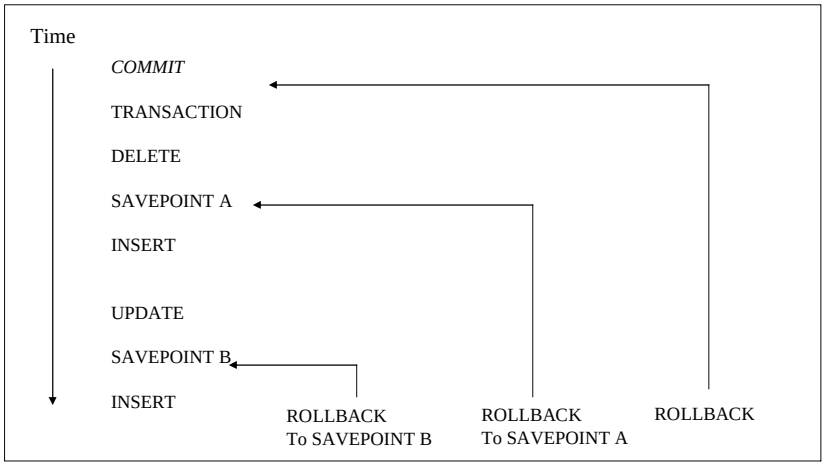
```
SELECT   AVG(commission_pct)
FROM     employees;
```

```
SELECT   AVG(NVL(commission_pct,0))
FROM     employees;
```

16

COMMIT & ROLLBACK statements…(1)

- Menjamin konsistensi data
- Menampilkan hasil perubahan sebelum membuat perubahan tersebut permanent



17

Data Types

Function	Sample
VARCHAR2*(size)*	Variable-length character data
CHAR[*(size)]*	Fixed-length character data
NUMBER[*(p,s)*]	Variabale-length numeric data
DATE	Date and time values
LONG	Variable-length character data up to 2 gigabytes
CLOB	Character data up to 4 gigabytes
RAW and LONG RAW	Raw binary data
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes
ROWID	Hexadecimal string representing the unique address of a row in its table

Constraints…(1)

18

- Constraints memaksa tabel atau kolom memenuhi aturan-aturan tertentu – sehingga data yang tidak valid tidak dapat masuk ke dalam tabel.
- Constraints menghalangi proses ‘delete’ jika terdapat pelanggaran ‘dependency’ / ketergantungan.
- Constraints dapat dibangun dengan cara:
  - Bersamaan pada saat tabel dibangun
  - Sesudah tabel dibangun

Constraints…(2)

19

Data Integrity Constraints

Constraint	**Keterangan**
NOT NULL	Memaksa kolom tidak boleh berisi nilai NULL
UNIQUE	Memaksa kolom atau kombinasi kolom harus memiliki nilai yang unik untuk setiap baris dalam tabel
PRIMARY KEY	Memaksa kolom harus memiliki nilai dan unik untuk setiap baris
FOREIGN KEY	Memaksa nilai yang dimasukan dalam kolom tertentu harus memiliki referensi di tabel lainnya
CHECK	Memaksa sebuah kondisi yang harus selalu benar

LOGIN…

Cara 1: Aktifkan software SQL Plus*:* *Start → Programs → Oracle-OraHome92 → Aplication Development → SQL Plus*

Masukkan username ***oraXX*** dengan password ***oracle***, kemudian klik login.

Ganti *XX* dengan nomor urut presensi anda.

Jika anda memiliki nomor urut presensi 24, maka:

Username : ora24
Password : oracle

Cara set tampilan:

```
SQL> COLUMN email FORMAT A50
SQL> COLUMN salary FORMAT 9999
```

Cara 2: Aktifkan browser (Internet Explorer), pada address bar ketikan:

http://oracle/isqlplus
↳ nama komputer server

1. Masukkan username *oraXX* dengan password *oracle*, kemudian klik login.
2. Setelah login berhasil, Anda dapat mengetikan semua perintah query pada *textarea enter statements* yang sudah tersedia.

*** Selamat belajar dan melangkah maju bersama ORACLE …….. ☺

PEMAHAMAN STUDI KASUS

1. Sebelum latihan dimulai, setiap peserta harus mengetahui tabel-tabel apa saja yang akan kita gunakan. Untuk itu, cobalah ketikan perintah query berikut pada *textarea Enter Statements* yang tersedia.

```
-- perintah untuk melihat semua tabel dan view milik user oraXXg
SELECT * FROM tab;
-- khusus untuk melihat tabel milik user oraXXg
SELECT * FROM user_tables;
-- melihat seluruh table, data dictionary, view atau dynamic view
SELECT * FROM all_tables;
```

2. Cek apakah ANDA mendapatkan tabel-tabel berikut:

COUNTRIES	JOB_HISTORY
DEPARTMENTS	LOCATIONS
EMPLOYEES	REGIONS
JOBS	

3. Untuk melihat struktur record dari sebuah tabel gunakan perintah DESCribe, misal:

```
DESC countries;
```

4. Untuk melihat *primary key* dan *foreign key* ataupun constraint lainnya, dapat menggunakan perintah sbb:

```
SELECT  constraint_name, constraint_type, search_condition, r_constraint_name
FROM    user_constraints
WHERE   table_name = 'COUNTRIES';
```

5. Jika ingin mengetahui arti kolom *constraint_name, constraint_type,* dan *r_constraint_name,* Anda dapat memintanya kepada ORACLE dengan memberikan perintah sbb:

```
SELECT  column_name, comments
FROM    dict_columns
WHERE   table_name = 'USER_CONSTRAINTS';
```

6. Jika ingin mendapatkan informasi tentang index atau constraint, Anda dapat memintanya pada tabel **USER_IND_COLUMNS** atau **USER_CONS_COLUMNS**.

```
SELECT  index_name, column_name, column_position
FROM    user_ind_columns
WHERE   table_name = 'EMPLOYEES';
```

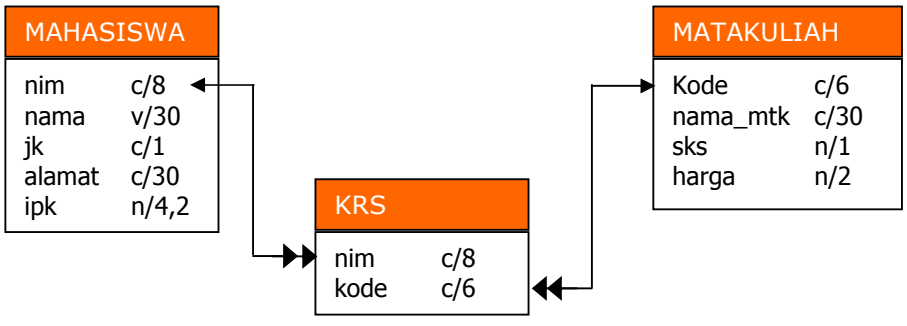
```
SELECT  *
FROM    user_cons_columns
WHERE   table_name = 'EMPLOYEES';
```

7. Cermati langkah 3 s/d 6, ulangi langkah-langkah tersebut untuk mendapatkan informasi dari seluruh tabel (countries, departments, employees, jobs, job_history, locations, regions).
   - Catat informasi jumlah record dari setiap tabel:

Nama Tabel	Jumlah Data *(Rows)*
COUNTRIES	
DEPARTMENTS	
EMPLOYEES	
JOBS	
JOB_HISTORY	
LOCATIONS	
REGIONS	

   - Gambarkan E-R diagramnya
   - Buatlah tabel contraint sesuai hasil pengamatan Anda

Contoh E-R diagram



Contoh Tabel Constraint

Table Name	Constraint_Name	Constraint_type	Search_Condition	Column_name	R_Constraint_Name
COUNTRIES	COUNTRY_ID_NN	C	"COUNTRY_ID" IS NOT NULL	COUNTRY_ID	
	COUNTRY_C_ID_PK	P		COUNTRY_ID	
	COUNTR_REG_FK	R		REGION_ID	REG_ID_PK

Basic SQL SELECT Statements

SQL
- Sebuah bahasa
- Standar ANSI (American National Standards Institute)
- Keyword tdk dapat disingkat
- Statement memanipulasi data dan mendefiniskan tabel dalam database

***i*SQL*Plus**
- Sebuah lingkungan untuk mengeksekusi statement SQL
- Milik Oracle
- Keyword dapat disingkat
- Command tidak mengijinkan memanipulasi nilai dalam database
- Berjalan menggunakan browser
- Tidak dapat diimplementasikan pada semua mesin

SQL statements		***i*SQL*Plus commands**

SELECT *|{[DISTINCT] *column*| *expression* [*alias*], …}
FROM *tables*;

- SELECT untuk mengidentifikasi kolom-kolom yang ditampilkan
- FROM untuk mengidentifikasi tabel yang digunakan
- SELECT, FROM adalah *keyword*
- SELECT employee_id, last_name, … adalah *clause*
- SELECT * FROM employees, adalah SQL *statement*

Selecting All Columns

```
SELECT  *   FROM departments;
```

Selecting Spesific Columns

```
SELECT  department_id, location_id
FROM    departments;
```

Writing SQL Statements
- SQL statements tidak case sensitive
- SQL statements dapat ditulis dalam satu atau banyak baris
- Keyword harus dituliskan secara berurutan
- Clause biasanya ditempatkan dalam satu baris
- Indent digunakan untuk memudahkan pembacaan

Column Heading Defaults
- *i*SQL*Plus
  - Default heading justification: Center
  - Default heading display: Uppercase
- SQL*Plus
  - Character dan Date column heading: Left-justified
  - Number column heading: Right-justified
  - Default heading display: Uppercase

Arithmetic Expressions

Operator	Description
*	Multiply
/	Divide
+	Add
-	Subtract

Defining a Column Alias

- Digunakan untuk memberi nama pada kolom
- Sangat berdayaguna pada proses kalkulasi
- Cara penulisan: bisa dengan menambahkan keyword AS antara nama kolom dan nama alias, tapi ini tidak mutlak
- Gunakan double quotionmarks jika mengandung spasi, spesial karakter atau case sensitive

Null Values

- Null adalah nilai yang tidak ada, kosong
- Tidak sama dengan **0** ataupun **spasi**, karena **0** adalah angka dan **spasi** adalah karakter

```
SELECT  last_name, salary, commission_pct,
        12*salary*commission_pct AS “Annual Salary”
FROM    employees;
```

Concatenation Operator

- Digunakan untuk menggabungkan kolom atau character string (diapit *single quotation marks*) ke kolom lain

```
SELECT  last_name || ‘ is a ‘ || job_id
        AS “Employee Details”
FROM    employees;
```

- Direpresentasikan dengan ||
- Hasil penggabungan ini berupa ekspresi character

LAB: Basic SQL SELECT Statements

1. Buatlah query yang dapat membantu ANDA untuk melengkapi tabel berikut ini:

Nama Tabel	Field_PK	Field_FK	Jumlah Data *(Rows)*
COUNTRIES			
DEPARTMENTS			
EMPLOYEES			
JOBS			
JOB_HISTORY			
LOCATIONS			
REGIONS			

2. Tampilkan informasi last name, job code, hire date dan employee number untuk setiap employee yang tercatat di tabel EMPLOYEES. <107 rows>

3. Buatlah sebuah query yang dapat menampilkan secara unik kode job dari tabel EMPLOYEES <19 rows>

4. Coba jalankan query berikut ini: <107 rows>

```
SELECT  employee_id, last_name, job_id, hire_date
FROM    employees;
```

   Modifikasikanlah query diatas, agar judul kolom dari hasil querynya menampilkan:

   Emp# Employee Job Hire Date

5. Tampilkan informasi tentang last_name, job dalam 1 kolom dan beri judul "Employees and Title" <107 rows>

6. Tampilkan informasi tentang variasi gaji yang diterima oleh para employee dan beri judul "Variasi Gaji" <57 rows>

Restricting and Sorting Data

```
SELECT      *|{[DISTINCT] column| expression  [alias], …}
FROM        tables
[WHERE      condition(s)]
[ORDER BY   {column, expr} [ASC|DESC]];
```

[**WHERE** *condition(s)*]

- Clause WHERE untuk membatasi *row(s)* yang ditampilkan
- Untuk tipe data: *character* dan *date* harus ditempatkan dalam *single quotation marks*

- *Character* bersifat *case sensitive*, *Date* bersifat *format sensitive*
- Default format untuk tipe *Date*: **DD-MON-RR**
- Alias tidak dapat digunakan pada clause WHERE
- Kondisi pembanding:

Operator	Arti
=	Sama dengan
>	Lebih besar
>=	Lebih besar atau sama dengan
<	Lebih kecil
<=	Lebih kecil atau sama dengan
<> atau != atau ^=	Tidak sama dengan
BETWEEN ... AND ...	Antara 2 nilai
IN(set)	Mencocokan suatu nilai dengan nilai dalam list Lawannya: NOT IN
LIKE	Mencocokan pola *character*
IS NULL	Uji nilai *null* Lawannya: IS NOT NULL

- Cara penulisan kondisi LIKE:
  - **%** merepresentasikan sejumlah karakter
  - **_** merepresentasikan satu karakter

  Gunakan **ESCAPE identifier** untuk mencari simbol % dan _ dalam data

Example:

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

```
..... WHERE     hire_date = '01-JAN-95'
..... WHERE     salary >= 1000
..... WHERE     last_name = 'Smith'
..... WHERE     salary BETWEEN 2500 AND 3500
..... WHERE     manager_id IN (100, 101, 201)
..... WHERE     last_name IN ('Hartstein', 'Vargas')
..... WHERE     manager_id IS NULL
..... WHERE     job_id LIKE '%SA\_%' ESCAPE '\'
```

- Clause WHERE dapat mengandung *Logical Conditions* **(AND, OR, NOT)***:*

```
..... WHERE   salary >= 10000 OR job_id LIKE '%MAN%'
```

[**ORDER BY** {column, expr} [ASC|DESC]]

- Default clause ORDER BY adalah ascending
- Dapat digunakan pada kolom alias

```
SELECT      employee_id, last_name, salary*12 annsal
FROM        employees
ORDER BY    annsal;
```

- Dapat dilakukan pada multi kolom

```
SELECT      last_name, department_id, salary
```

```
FROM        employees
ORDER BY    department_id, salary DESC;
```

- Dapat juga dilakukan SORT berdasarkan kolom yang tidak ditampilkan dalam daftar SELECT.

LAB: Restricting and Sorting Data

1. Buatlah query untuk menampilkan last name dan salary dari employee yang gajinya lebih kecil dari $10,000.

2. Buatlah query untuk menampilkan last name, first name dan department number untuk employee number 176.

3. Pahami output yang dihasilkan oleh query berikut ini:

```
SELECT TO_CHAR(
     ADD_MONTHS(hire_date,1),
     'DD-MON-YYYY') "Next month"
     FROM employees
     WHERE last_name = 'Baer';
```

4. Modifikasi query no.3 untuk menampilkan hire_date, hire_date+1bln dan hire_date+2bln dari employees yang bernama (last_name) 'Tucker' dan 'Atkinson' Buatlah kesimpulan tentang efek dari fungsi ADD_MONTHS.

5. Modifikasi query no.1 agar dapat menampilkan semua karyawan yang memiliki gaji antara $5,000 sampai dengan $12,000

6. Buatlah query yang dapat menampilkan last name, job id dan hire date dari employee yang mulai bekerja (hire) antara tanggal 20 Februari 1998 sampai dengan 1 Mei 1998.

7. Tampilkan last name dan department number untuk semua employees dalam department 20 sampai 50 dalam suatu urutan berdasarkan last name.

8. Modifikasi query no. 5 sehingga menampilkan semua employees dalam department 20 sampai 50 dan juga memiliki gaji antara $5,000 sampai dengan $12,000.

9. Tampilkan last name dan hire date untuk setiap employee yang dipekerjakan tahun 1994.

10. Tampilkan last name dan job id dari semua employee yang tidak memiliki manager.

11. Tampilkan last name, salary dan commission untuk semua employee yang mendapat commission saja. Urutkan secara descending berdasarkan salary dan commission.

12. Tampilkan last name, dari semua employee yang huruf kedua dari last name-nya adalah o.

13. Tampilkan last name, job dan salary untuk semua employee yang memiliki job sebagai sales representative atau stock clerk dan memiliki gaji tidak sama dengan $2,500, $3,500 atau $7,500

☆ 14. Tampilkan informasi employee yang memiliki id bilangan ganjil

☆ 15. Jika diasumsikan bahwa setiap bulan ada 23 hari kerja, hitunglah berapa gaji sehari dari setiap karyawan

Conversion Functions

TO_CHAR(date, 'format_model'**)**
TO_CHAR(number, 'format_model'**)**
TO_NUMBER(char[, 'format_model'**])**
TO_DATE(char[, 'format_model'**])**

Format model:

- harus dituliskan dalam *single quotation marks* dan bersifat *case sensitive*

Element	Deskripsi
Date Format Model	
SCC or CC	Century, server prefixes B.C. date with -
YYY or YY or Y	Last three, two, or one digits of year
Y,YYY	Year with comma in this position
IYYY, IYY, IY, I	Four, three, two, or one digit year based on the ISO standard
SYEAR or YEAR	Year spelled out; server prefixes B.C. dat with -
Q	Quarter of year
MM	Month: two-digit value
MONTH	Name of month padded with blanks to length of nine characters
MON	Name of month, three-letter abbreviation
WW or W	Week of year or month
DDD or DD or D	Day of year, month, or week
DAY	Name of day padded with blanks to a length of nine characters
DY	Name of day; three-letter abbreviation
Time Format Model	
AM or PM or A.M. or P.M.	Meridian indicator
HH or HH12 or HH24	Hour of day, or hour(1-12), or hour(0-23)
MI	Minute (0-59)
SS	Second (0-59)
Other Formats	
/ . ,	Punctuation is reproduced in the result
"ot the"	Quoted string is reproduced in the result
Specifying Suffixes to Influence Number Display	
TH	Ordinal number (for example, DDTH for 4^{TH})
SP	Spelled-out number (for example, DDSP for FOUR)
SPTH or THSP	Spelled-out ordinal numbers (for example, DDSPTH for FOURTH)

Number format model	
9	Represents a number
0	Forces a zero to be displayed
$	Places a floating local currency symbol
.	Prints a decimal point
,	Prints a thousand indicator

Example: cobalah jalankan SQL statement di bawah ini dan pahami maksudnya

```
SELECT    last_name, TO_CHAR(hire_date,
              'fmDdspth "of" Month YYY fmHH:MI:SS AM') HIREDATE
FROM      employees;

SELECT    TO_CHAR(salary, '$99,999.00') SALARY
FROM      employees
WHERE     last_name= 'Ernst';

SELECT    last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM      employees
WHERE     hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

- RR date format

		If the spesified two-digit year is:	
		0-49	**50-99**
If two digits of the current year are:	**0-49**	The return date is in the **current century**	The return date is in the **century before** the current one
	50-99	The return date is in the **century after** the current one	The return date is in the **current century**

Current Year	Specified Date	RR Format DD-MON-RRRR	YY Format DD-MON-YYYY
1995	27-OCT-95		
1995	27-OCT-17		
2001	27-OCT-17		
2001	27-OCT-95		

Lengkapi tabel ini...

General Functions

NVL (expr1, expr2**)** mengubah NULL menjadi nilai actual

NVL2 (expr1, expr2, expr3**)** jika expr1 <> NULL maka NVL2 mengembalikan nilai expr2, jika expr1 = NULL maka NVL2 mengembalikan nilai expr3.

NULLIF (expr1, expr2**)** membandingkan expr1 dan expr2 --- kembalikan NULL jika sama, kembalikan expr1 jika tidak sama

COALESCE (expr1, expr2, ..., exprn**)** mengembalikan expr yang bukan NULL yang pertama kali ditemukan dalam expr list.

Conditional Expressions

- Memungkinkan kita menggunakan logika IF-THEN-ELSE dalam SQL statement
- Tersedia 2 metode:
  - CASE expression
  - DECODE function
- Bentuk umum CASE:

```
CASE expr   WHEN comparison_expr1  THEN return_expr1
            [WHEN comparison_expr2 THEN return_expr2
            [WHEN comparison_exprn THEN return_exprn
            ELSE else_expr]
END
```

- Bentuk umum DECODE:

```
DECODE(col/expression, search1, result1
                       [, search2, result2,...,]
                       [, default])
```

Example:

```
SELECT   last_name, job_id, salary,
         CASE job_id     WHEN 'IT_PROG'    THEN 1.10*salary
                         WHEN 'ST_CLERK'   THEN 1.15*salary
                         WHEN 'SA_REP'     THEN 1.20*salary
         ELSE salary END  "REVISED_SALARY"
FROM     employees;

SELECT   last_name, job_id, salary,
         DECODE (job_id,   'IT_PROG',    1.10*salary,
                           'ST_CLERK',   1.15*salary,
                           'SA_REP',     1.20*salary,
                 salary) "REVISED_SALARY"
FROM     employees;
```

LAB: Functions & Expressions

1. Tulis query tanggal server sekarang
2. Tulis query untuk menampilkan usia anda saat ini (dalam hari dan dalam bulan)

3. Untuk setiap employee, tampilkan employee number, last name, salary dan salary baru yang merupakan perkalian 15% dengan salary yang lama dan tampilkan dalam suatu nilai bulat. Tampilkan alias untuk tiap kolom tersebut dengan “Emp#”, “Name”, “Salary”, “New Salary”

4. Tuliskan query yang menampilkan last name employee dengan huruf pertamanya besar dan lainnya kecil, juga tampilkan panjang tiap last name yang ditampilkan tersebut untuk seluruh employee yang last name memiliki huruf depan J, A atau M.

5. Buatlah sebuah query yang menampilkan last name dan hitunglah selisih bulan antara saat ini dengan hire date. Berilah label "Bulan Kerja". Urutkan hasil yang akan ditampilkan berdasar selisih bulan secara ascending.

6. Buatlah sebuah query yang menghasilkan tampilan dengan format berikut untuk tiap employees

   <last name> earns **<salary>**monthly but wants **<3 times salary>**

7. Tampilkan last name, salary, commission_pct, annual salary dan informasi income dari seluruh employees dengan ketentuan sbb:
   - jika commission_pct <> NULL maka income= salary+(salary*commission_pct)
   - jika commission_pct = NULL maka income= salary

8. Tampilkan first_name, panjang(first_name), last_name, panjang(last_name), result dengan ketentuan sbb:
   - jika panjang(first_name) = panjang(last_name) maka result = NULL
   - jika panjang(first_name) <> panjang(last_name) maka result = panjang(first_name)

9. Apakah maksud dari query berikut ini?

```
SELECT    last_name, salary, commission_pct,
          NVL2(commission_pct, 'SAL+COMM', 'SAL') income
FROM      employees
WHERE     department_id IN (50, 80);
```

10. Apakah maksud dari query berikut ini?

```
SELECT    last_name, commission_pct, salary,
          COALESCE(commission_pct, 10, salary) COMM
FROM      employees
ORDER BY  commission_pct;
```

11. Buatlah query yang dapat menampilkan seperti contoh berikut ini untuk seluruh employees

LAST_NAME	SALARY
King	*******24000
Kochhar	********7000
De Haan	********7000
Hunold	********9000
Ernst	********6000

12. Buatlah query untuk menampilkan last name dan jumlah commission dari seluruh employee. Jika seorang employee tidak dapat commission, maka tampilkan tulisan "No Commission". Berilah label "COMM" untuk kolom ini.

13. Dengan menggunakan fungsi DECODE, tuliskan query yang dapat menampilkan level semua employee berdasarkan pada nilai JOB_ID dengan ketentuan sbb:

JOB	GRADE
AD_PRESS	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of above	0

14. Tulis ulang query no. 12, namun dengan menggunakan perintah CASE.

Displaying Data from Multiple Tables

1. **Cartesian Product:** Semua baris dari tabel pertama dijoinkan terhadap semua baris dari tabel kedua

A

Q: SELECT last_name
FROM employees

R: ..... rows selected

B

Q: SELECT department_name
FROM departments

R: ..... rows selected

C

Q: SELECT last_name, department_name dept_name
FROM employees, departements

R: ..... rows selected

2. **JOIN:**
   - digunakan ketika kita memerlukan data yang bersumber pada lebih dari satu tabel.
   - untuk melakukan JOIN, perhatikan kolom yang menjadi **Primary Key** dan **Foreign Key** dari setiap tabel yang ingin di-**JOIN**-kan.
   - gunakan nama tabel untuk mempertegas pengambilan kolom, terutama jika nama kolom yang sama terdapat pada lebih dari satu tabel.
   - Gunakan tabel alias untuk menyederhanakan penulisan query dan meningkatkan unjuk kerja.

➢ Equijoin (juga biasa disebut dengan istilah: simple join or inner join)

Try it...

```
SELECT    e.employee_id, e.last_name, e.department_id,
          d.department_id, d.location_id
FROM      employees e, departments d
WHERE     e.department_id = d.department_id;

SELECT    e.last_name, d.department_name, l.city
FROM      employees e, departments d, locations l
WHERE     e.department_id = d.department_id
          AND d.location_id = l.location_id;
```

➢ Nonequijoin: sebuah join yang dibentuk dari operator selain ‘sama dengan’

```
CREATE TABLE job_grades (grade_level varchar2(3),
                         lowest_sal  number(10),
                         highest_sal number(10));
```

Masukkan data berikut dalam tabel job_grades:

grade_level	lowest_sal	highest_sal
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

Cara input:

```
INSERT INTO job_grades VALUES ('A', 1000, 2999);
```

- Tampilkan struktur & data dari tabel EMPLOYEES
- Tampilkan struktur & data dari tabel JOB_GRADES
- Buatlah query untuk menampilkan last_name, salary dan grade_level dari setiap EMPLOYEES.

- Outer Join:

```
SELECT   e.last_name, e.department_id, d.department_name
FROM     employees e, departments d
WHERE    e.department_id = d.department_id;
```

R: ..... rows selected

```
SELECT   e.last_name, e.department_id, d.department_name
FROM     employees e, departments d
WHERE    e.department_id(+) = d.department_id;
```

R: ..... rows selected

```
SELECT   e.last_name, e.department_id, d.department_name
FROM     employees e, departments d
WHERE    e.department_id = d.department_id(+);
```

R: ..... rows selected

- Jelaskan satu persatu apa perbedaan dari 3 variasi query diatas

- Self Join:

Try it...

```
SELECT   worker.last_name || ' works for ' || manager.last_name
FROM     employees worker, employees manager
WHERE    worker.manager_id=manager.employee_id;
```

- Cross Join (sama dengan Cartesian product):

Try it...

```
SELECT   last_name, department_name
FROM     employees
CROSS JOIN departments;
```

- Natural Join

Try it...

```
SELECT   department_id, department_name, location_id, city
FROM     departments
NATURAL JOIN locations
WHERE    department_id IN (20, 50);
```

Buat catatan yang ANDA anggap penting tentang keenam type JOIN:…

AGGREGATING DATA using Group Functions

Group Functions:

Function	Keterangan
AVG([DISTINCT | ALL] n)	Menghitung dari-rata dari sekelompok data dengan mengabaikan nilai *null*
COUNT({* | DISTINCT | ALL] expr})	Menghitung banyak rows dari suatu ekspresi tertentu
MAX([DISTINCT | ALL] expr)	Menghitung nilai maximum, abaikan nilai *null*
MIN([DISTINCT | ALL] expr)	Menghitung nilai minimum, abaikan nilai *null*
STDDEV([DISTINCT | ALL] x)	Menghitung standard deviasi, abaikan nilai *null*
SUM([DISTINCT | ALL] n)	Menghitung jumlah dari n, abaikan nilai *null*
VARIANCE([DISTINCT | ALL] x)	Variance dari n, abaikan nilai *null*

LAB: Aggregating

- Tampilkan informasi tentang salary rata-rata, salary maximun, salary minimum dan salary total dari seluruh karyawan yang job_id nya: SA_REP, MK_REP, HR_REP, PR_REP

- Tampilkan informasi tentang jumlah karyawan yang bekerja di department 50

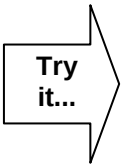
☆ - Tampilkan informasi tentang employee yang berpenghasilan tertinggi dan terendah. **(Penghasilan = gaji + bonus)**

- Perhatikan 2 query berikut, jelaskan perbedaan output yang dihasilkan!

```
SELECT   AVG(commission_pct)
FROM     employees;

SELECT   AVG(NVL(commission_pct, 0))
FROM     employees;
```

GROUP BY clause

```
SELECT      column, group_function(column)
FROM        table
[WHERE      conditions]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

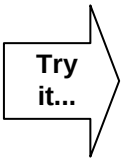


- Cobalah untuk menampilkan rata-rata salary dari setiap department_id
- Perbaiki query sebelumnya agar output yang dihasilkan urut mulai dari rata-rata salary terendah

HAVING clause

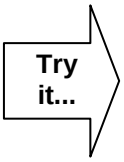
```
SELECT      column, group_function(column)
FROM        table
[WHERE      conditions]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

- Jalankan query berikut:




```
SELECT      department_id, COUNT(last_name)
FROM employees;
```

Jelaskan, kenapa query tersebut ERROR?




```
SELECT      department_id, AVG(salary)
FROM employees
WHERE       AVG(salary) >8000
GROUP BY    department_id;
```

- Perbaiki query diatas agar bisa menghasilkan output yang diinginkan!

LAB: Displaying Data from Multiple

1. Buatlah query untuk menampilkan last name, department id dan department name untuk seluruh karyawan.

2. Buatlah query untuk menampilkan last name, department name, location ID, dan city dari semua karyawan yang mendapat commission.

3. Tampilkan informasi tentang last name dan department name dari karyawan yang last name-nya mengandung huruf ‘a’ (lowercase).

4. Tampilkan last name, jod, department id, dan department name untuk seluruh karyawan yang bekerja di Toronto.

5. Tampilkan informasi tentang employee name, employee id, manager name, manager id. Beri label kolom sbb: Employee, Emp#, Manager, Mgr#

6. Apa maksud dari query berikut ini?

```
SELECT e.department_id department, e.last_name employee, c.last_name colleague
FROM   employees e JOIN employees c
ON     (e.department_id = c.department_id)
WHERE  e.employee_id <> c.employee_id
ORDER BY e.department_id, e.last_name, c.last_name
```

LAB: AGGREGATING DATA using Group

1. Tampilkan informasi tentang minimun, maximum, total, dan rata-rata salary dari setiap jod_id. Gunakan fungsi ROUND untuk membuat tampilan tanpa digit dibelakang koma). Beri judul setiap kolom: “Job”, “Maximum”, “Minimum”, “Sum”, “Average”.

2. Tampilkan informasi tentang jumlah orang yang berkedudukan sebagai manager, tanpa perlu menampilkan daftarnya. Judul kolom output diberi judul: ‘Number of Managers’. Petunjuk: gunakan kolom ‘manager_id’ untuk identifikasi.

☆ 3. Tampilkan informasi tentang jumlah employee pada setiap variasi gaji beri judul "Variasi Gaji" dan “Jumlah Karyawan” <57 rows>

4. Apa arti dari query berikut ini:

```
SELECT   COUNT(*) total,
         SUM(DECODE(TO_CHAR(hire_date, ‘YYYY’), 1995,1,0)) “1995”,
         SUM(DECODE(TO_CHAR(hire_date, ‘YYYY’), 1996,1,0)) “1996”,
         SUM(DECODE(TO_CHAR(hire_date, ‘YYYY’), 1997,1,0)) “1997”,
         SUM(DECODE(TO_CHAR(hire_date, ‘YYYY’), 1998,1,0)) “1998”
FROM     employees;


SELECT   job_id “Job”,
         SUM(DECODE(department_id, 20, salary)) “Dept 20”,
         SUM(DECODE(department_id, 50, salary)) “Dept 50”,
         SUM(DECODE(department_id, 80, salary)) “Dept 80”,
         SUM(DECODE(department_id, 90, salary)) “Dept 90”,
         SUM(salary) “Total”
FROM     employees
GROUP BY job_id;
```

SUBQUERIES

Main Query:

Siapa saja employee yang punya salary lebih besar dari Abel?

Subquery:

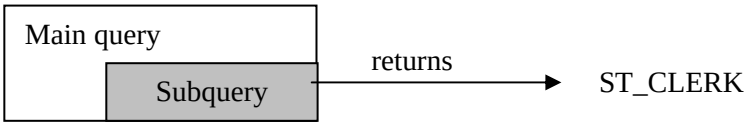
Berapakah salary Abel?

- Subquery (inner query) dijalankan sebelum main query
- Output dari subquery digunakan untuk mengendalikan output dari main query (outer query)
- Contoh:

```
SELECT    last_name
FROM      employees          11000
WHERE     salary >
                  (SELECT     salary
                   FROM       employees
                   WHERE      last_name = 'Abel');
```

Types of Subqueries

- Single-row subquery:



  - hanya mengembalikan 1 nilai
  - menggunakan operator pembanding : {=, >, >=, <, <=, <>}
  - contoh:

A

```
SELECT    last_name, job_id, salary
FROM      employees
WHERE     job_id =          ST_CLERK
                  (SELECT     job_id
                   FROM       employees
                   WHERE      employee_id = 141)
                                2600
AND       salary >
                  (SELECT     salary
                   FROM       employees
                   WHERE      employee_id = 143);
```

B

```
SELECT    last_name, job_id, salary
FROM      employees
                                2100
WHERE     salary >
                  (SELECT     MIN(salary)
                   FROM       employees);
```

Using Group Function

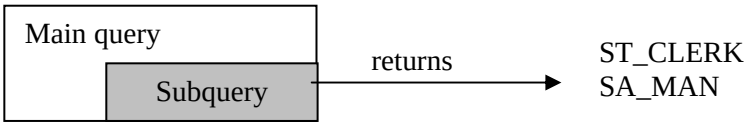
C

```
SELECT      department_id, MIN(salary)
FROM        employees
GROUP BY    department_id
HAVING      MIN(salary) > (SELECT  MIN(salary)
                           FROM    employees
                           WHERE department_id = 50);
```

2100

Using HAVING clause

- Multiple-row subquery



  - mengembalikan lebih dari satu baris
  - menggunakan operator pembanding : {IN, ANY, ALL}
  - contoh:

```
SELECT    employee_id, last_name, job_id, salary
FROM      employees
WHERE     salary < ANY (SELECT    salary
                        FROM      employees
                        WHERE     job_id = 'IT_PROG')
AND       job_id <>  'IT_PROG';
```

9000, 6000, 4800, 4800, 4200

catatan: **< ANY .... lebih kecil dari 9000 (max)**
> ANY .... lebih besar dari 4200 (min)

LAB: Subqueries

o Buat query untuk menampilkan name dan hire date dari setiap employee yang department-nya sama dengan Zlotkey, kecuali Zlotkey. --- output: 33 rows

o Buat query untuk menampilkan employee number, last name dan salary dari seluruh employee yang mendapatkan salary diatas rata-rata. Tampilkan urut berdasarkan salary. --- output: 51 rows

o Tampilkan last name, department number dan job ID dari seluruh employee yang department location ID-nya adalah 1700. --- output: 18 rows

o Tampilkan last name dan salary dari setiap orang yang mempunyai manager bernama King. --- output: 14 rows

o Tampilkan department number, last name dan job ID dari setiap employee yang bekerja di department Executive. --- output: 3 rows

o Tampilkan informasi tentang jumlah orang yang bekerja pada setiap jenis pekerjaan. --- output: 12 rows

Substitution Variables

Gunakan variabel substitusi pada *i*SQL*Plus untuk:

- memberikan nilai secara temporary:
  - Single ampersand (&)
  - Double ampersand (&&)
  - DEFINE command

- mengirimkan nilai variabel antar SQL statement
- Merubah headers dan footers secara dinamis
- Contoh:

```
SELECT    employee_id, last_name, salary, department_id
FROM      employees
WHERE     employee_id = &employee_num;
```

.... coba isi dengan: 101

```
SELECT    last_name, department_id, salary*12
FROM      employees
WHERE     job_id = '&job_title';
```

.... coba isi dengan: IT_PROG

..... coba ganti WHERE dengan:

```
WHERE  job_id = UPPER('&job_title');
```

..... buat catatan perbedaan antar kedua statement WHERE tersebut.

```
SELECT      employee_id, last_name, job_id, &column_name
FROM        employees
WHERE       &condition
ORDER BY  &order_column;
```

.... coba isi dengan:

```
column_name      : salary
condition        : salary > 15000
order_column     : last_name
```

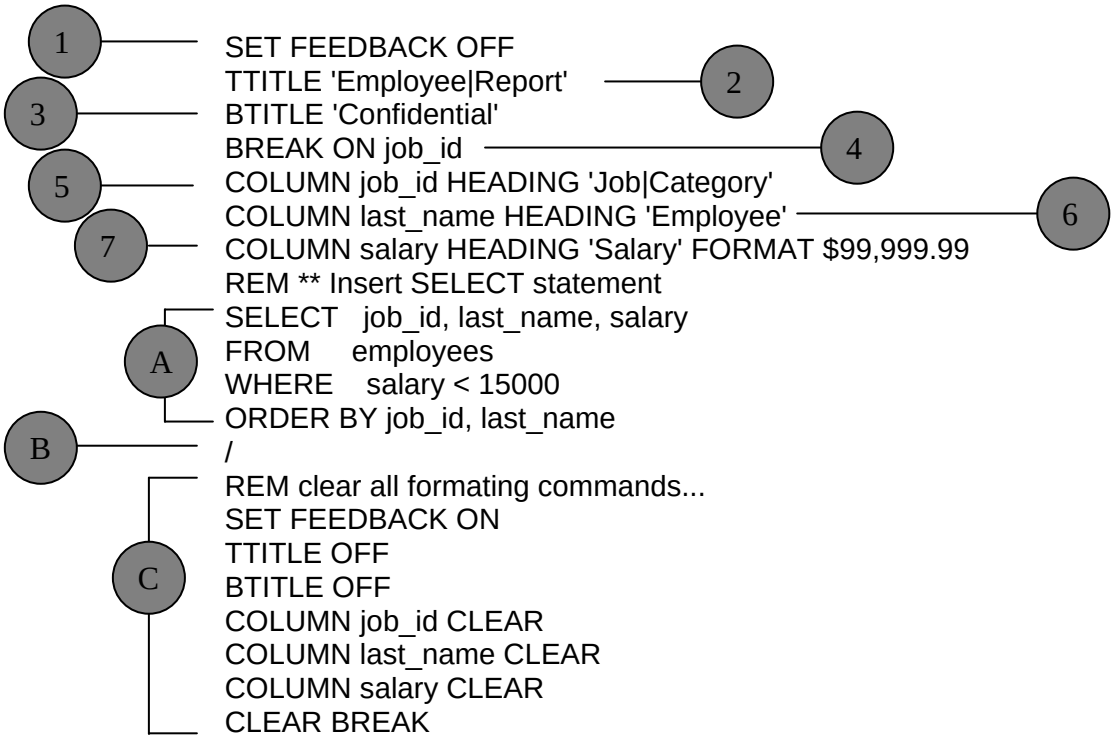
```
DEFINE      employee_num = 200

SELECT      employee_id, last_name, salary, department_id
FROM        employees
WHERE       employee_id = &employee_num;
```

- Apakah fungsi dari perintah DEFINE?

```
SELECT      employee_id, last_name, job_id, &&column_name
FROM        employees
ORDER BY  &column_name;
```

- Apakah maksud dari variabel yang menggunakan double-ampersand seperti pada query diatas?

Creating a Script File to Run a Report



Langkah-langkah pemahaman:

1. Tuliskan kelompok perintah A, lakukan **execute**, cermati hasilnya
2. Tambahkan perintan 1, lakukan **execute**, cermati perubahannya

Ulangi langkah 2, tambahkan satu persatu baris perintah 2 s/d 7.

Ingat!!! Setiap penambahan 1 baris, lakukan **execute**.

3. Simpan script anda, dan beri nama file **script_01**.
4. Bersihkan layar, click tombol " **Clear Screen**"
5. Eksekusilah sql statement berikut:

```
SELECT      *
FROM        employees
ORDER BY    job_id, last_name
```

   Cermati apakah tampilan sesuai dengan yang diharapkan?

6. Buka kembali **script_01,** lalu tambahkan dengan kelompok perintah B dan C, lakukan **execute**.
7. Kembali jalankan sql statement di langkah 5 dan cermati perubahan apa yang terjadi?

Buatlah catatan arti dari perintah:

- SET FEEDBACK ON|OFF
- TTITLE ON|OFF
- BTITLE ON|OFF
- COLUMN column_name HEADING heading_column
- BREAK ON column_name
- CLEAR BREAK

Jalankan

```
BREAK ON DEPARTMENT_ID SKIP 1 ON JOB_ID SKIP 1 DUPLICATES
COMPUTE SUM OF SALARY ON DEPARTMENT_ID
COMPUTE AVG OF SALARY ON JOB_ID
SELECT DEPARTMENT_ID, JOB_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID IN ('SH_CLERK', 'SA_MAN')
AND DEPARTMENT_ID IN (50, 80)
ORDER BY DEPARTMENT_ID, JOB_ID;
```

LABS: Creating a Script File to Run a Report

1. Buatlah sebuah script dengan memodifikasi query yang tersedia sehingga menghasilkan output seperti pada tampilan dibawah.

```
SELECT d.department_name, e.last_name, e.hire_date, e.salary, e.salary*12 asal
FROM   departments d, employees e, locations l
WHERE  e.department_id = d.department_id
AND    d.location_id   = l.location_id
AND    l.city          = 'Seattle'
ORDER BY d.department_name
```

DEPARTMENT NAME	EMPLOYEE NAME	START DATE	SALARY	ANNUAL SALARY
Accounting	Higgins	07-JUN-94	$12,000.00	$144,000.00
	Gietz	07-JUN-94	$8,300.00	$99,600.00
Administration	Whalen	17-SEP-87	$4,400.00	$52,800.00
Executive	King	17-JUN-87	$24,000.00	$288,000.00
	Kochhar	21-SEP-89	$17,000.00	$204,000.00
	De Haan	13-JAN-93	$17,000.00	$204,000.00
Finance	Greenberg	17-AUG-94	$12,000.00	$144,000.00
	Faviet	16-AUG-94	$9,000.00	$108,000.00
	Chen	28-SEP-97	$8,200.00	$98,400.00
	Sciarra	30-SEP-97	$7,700.00	$92,400.00
	Urman	07-MAR-98	$7,800.00	$93,600.00
	Popp	07-DEC-99	$6,900.00	$82,800.00
Purchasing	Raphaely	07-DEC-94	$11,000.00	$132,000.00
	Khoo	18-MAY-95	$3,100.00	$37,200.00
DEPARTMENT NAME	EMPLOYEE NAME	START DATE	SALARY	ANNUAL SALARY
	Baida	24-DEC-97	$2,900.00	$34,800.00
	Tobias	24-JUL-97	$2,800.00	$33,600.00
	Himuro	15-NOV-98	$2,600.00	$31,200.00
	Colmenares	10-AUG-99	$2,500.00	$30,000.00

2. Modifikasi script diatas, sehingga dapat menampilkan output untuk city lainnya secara dinamis. Gunakan variabel substitusi!!!

3. Modifikasi query untuk menampilkan informasi tentang karyawan dan managernya sehingga bisa dikeluarkan informasi daftar karyawan untuk manager tertentu saja.

```
SELECT   worker.last_name || ' works for ' || manager.last_name
         FROM          employees worker, employees manager
         WHERE         worker.manager_id=manager.employee_id;
```

4. Buatlah script seperti dibawah ini dan cermati hasilnya.

```
BREAK ON DEPARTMENT_ID SKIP 1 ON JOB_ID SKIP 1 DUPLICATES
COMPUTE SUM LABEL ‘Jumlah‘ OF SALARY ON DEPARTMENT_ID
COMPUTE AVG LABEL ‘Rerata‘ OF SALARY ON JOB_ID
SELECT DEPARTMENT_ID, JOB_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID IN ('SH_CLERK', 'SA_MAN')
AND DEPARTMENT_ID IN (50, 80)
ORDER BY DEPARTMENT_ID, JOB_ID;
```

5. Buatlah script untuk menampilkan department_id, last_name, salary dari setiap employee yang dikelompokan menurut department_id-nya. Tampilkan juga total salari dari setiap department.

Langkah Persiapan…

Buatlah beberapa tabel yang akan digunakan dalam latihan ini:

- C_EMPS dicopy dari tabel employees
- C_DEPTS dicopy dari tabel departments

Contoh:
```
CREATE TABLE c_emps
AS
  SELECT *
  FROM employees;
```

Adding a New Row to a Table

INSERT INTO *table* [(*column* [, *column*…])]
VALUES (*value* [, *value*…]);

1
```
INSERT INTO c_depts (department_id, department_name,
                                    manager_id, location_id)
VALUES      (70, 'Public Relations', 100, 1700);
```

*** Gunakan *single quotation marks* untuk insert data bertipe *character* & *date*

Methods for Inserting Null Values

1. **Explicit Method:** menuliskan kolom yang akan diberi nilai pada daftar kolom, sedang kolom yang tak bernilai tidak perlu disebutkan.

2
```
INSERT INTO c_depts (department_id, department_name)
VALUES      (30, 'Purchasing');
```

2. **Implicit Method:** tidak perlu menuliskan nama-nama kolom pada , daftar kolom, tapi ketika menuliskan value kita harus memberi nilai pada setiap kolom sesuai urutan struktur tabel.

3
```
INSERT INTO c_depts
VALUES      (100, 'Finance', NULL, NULL);
```

Inserting Special Values

4
```
INSERT INTO c_emps (employee_id,
                    first_name, last_name,
                    email, phone_number,
                    hire_date, job_id, salary,
                    commission_pct, manager_id,
                    department_id)
```

```
VALUES      (113, '
             'Louis', 'Popp',
             'LPOPP', '515.124.4567',
             SYSDATE, 'AC_ACCOUNT', 6900,
             NULL, 205, 100);
```

5

```
INSERT INTO c_emps
VALUES      (114, '
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'AC_ACCOUNT', 11000, NULL, 100, 30);
```

Creating a Script

6

```
INSERT INTO c_depts
            (department_id, department_name, location_id)
VALUES      (&department_id, '&department_name', &location);
```

*** Coba isi data: 40, Human Resource, 2500
*** Cek apakah data sudah masuk kedalam tabel

Copying Rows from Another Table

7

```
INSERT INTO sales_reps (id, name, salary, commission_pct)
  SELECT    employee_id, last_name, salary, commission_pct
  FROM      employees
  WHERE     job_id LIKE '%REP%';
```

*** Sebelum jalankan SQL statement diatas, buat terlebih dahulu tabel sales_reps.

```
CREATE TABLE sales_reps
                    (id  NUMBER(6),
                     name  VARCHAR2(25),
                     salary  NUMBER(8,2),
                     commission_pct  NUMBER(2,2));
```

The UPDATE Statement Syntax

```
UPDATE      table [(column [, column…])]
SET         column = value [, column = value, …]
[WHERE      condition];
```

8

```
UPDATE   c_emps
SET      departement_id = 70
WHERE    employee_id = 113;
```

9

```
UPDATE   c_emps
SET      job_id     = (SELECT    job_id
                       FROM      employees
                       WHERE     employee_id = 205),
         salary     =  (SELECT   salary
```

updating two columns with a subquery

```
                    FROM    employees
                    WHERE   employee_id = 205)
WHERE   employee_id = 114;
```

10

```
UPDATE  c_emps
SET     department_id = (SELECT     department_id
                    FROM    employees
                    WHERE   employee_id = 100)
WHERE   job_id      = (SELECT   job_id
                    FROM    employees
                    WHERE   employee_id = 200);
```

updating rows based on another table

11

```
UPDATE  employees
SET     department_id = 55
WHERE   department_id = 110;
```

Buat catatan tentang error yang terjadi, kenapa? Untuk membantu pemahaman ANDA, coba lakukan update tersebut pada tabel c_emps.

The DELETE Statement

DELETE [FROM] *table*
[WHERE condition];

12

```
DELETE FROM c_depts
WHERE   departement_name = 'Finance';
```

13

```
DELETE FROM c_emps;
```

14

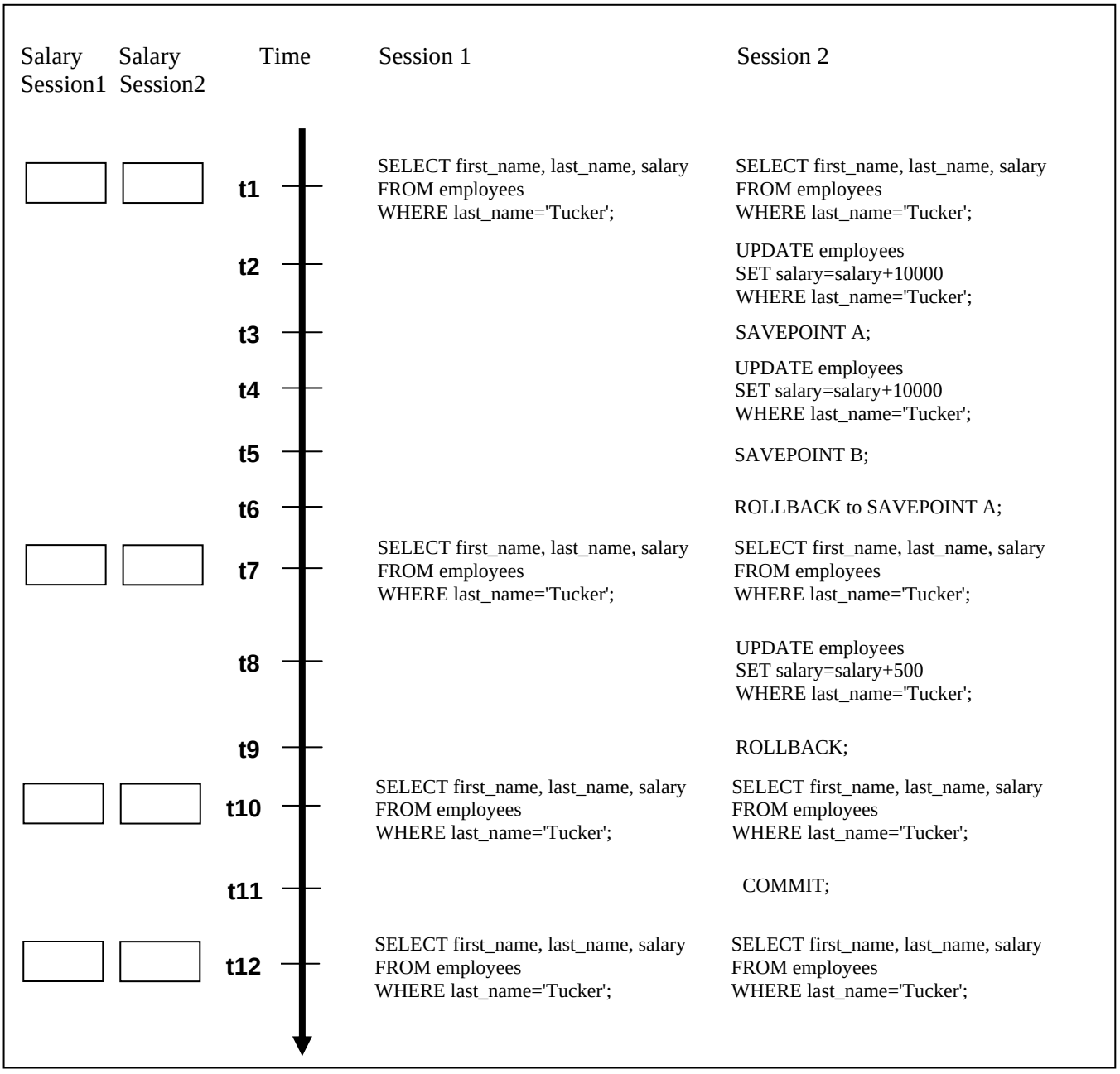
```
DELETE FROM c_emps
WHERE   department_id =
        (SELECT  department_id
          FROM    departments
          WHERE  department_name LIKE '%Public%');
```

delete rows based on another table

15 Cobalah buat SQL statement untuk menghapus record yang **department_id** adalah **30** dan **40** dari tabel **departments**

Transactions Controls

Bukalah dua sesi pada komputer ANDA, seakan-akan ada 2 user yang bekerja atas data yang sama. Tujuan percobaan dibawah ini adalah untuk memberikan pemahaman tentang statement ***Commit, Rollback dan Savepoint***.



Database Objects

Object	Description
Table	Menyimpan data
View	Subset data yang berasal dari satu tabel atau lebih
Sequence	Pembangkit nilai numeric
Index	Meningkatkan unjuk kerja dari beberapa query
Synonym	Memberikan alternatif nama pada *obyek*

- Aturan Penamaan untuk NAMA TABEL dan NAMA KOLOM
  - Harus diawali dengan sebuah huruf
  - Panjang nama 1-30 character
  - Hanya boleh mengandung character: A - Z, a - z, 0 - 9, _ (underscore), $, dan #
  - Tidak boleh terjadi duplikasi nama untuk *object* yang dimiliki oleh user yang sama
  - Tidak menggunakan kata-kata yang sudah digunakan Oracle Server

- CREATE TABLE statement
  - User harus memiliki hak (CREATE TABLE privilege)
  - User harus memiliki *storage area*

1

```
CREATE TABLE   dept
               (deptno   NUMBER(2),
                dname    VARCHAR2(14),
                loc      VARCHAR2(13));
```

- Tables dalam ORACLE DATABASE
  - User tables:
    - merupakan kumpulan tabel yang dibangun dan dirawat oleh user
    - berisi informasi dari user
  - Data dictionary:
    - merupakan kumpulan tabel yang dibangun dan dirawat oleh Oracle Server
    - berisi informasi tentang database

Cobalah jalankan dan buat catatan maksud dari setiap SQL statement berikut:

2

```
SELECT     table_name
FROM user_tables;

SELECT DISTINCT object_type
FROM user_objects;

SELECT *
FROM user_catalog
```

- Creating a Table by Using a Subquery

Jika tidak diberi nama alias akan terjadi error

3

```
CREATE TABLE dept80
AS
    SELECT    employee_id, last_name, salary*12  ANNSAL, hire_date
    FROM      employees
    WHERE     department_id = 80
```

- ALTER TABLE statement

  Digunakan untuk:
  - menambahkan kolom baru
  - modifikasi kolom yang sudah ada
  - mendefinisikan nilai *default* untuk kolom baru
  - membuang kolom *(drop)*

4

Cermati perubahan yang terjadi dari setiap SQL statemen dibawah ini:

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));

ALTER TABLE dept80
MODIFY    (last_name VARCHAR2(30));

ALTER TABLE dept80
DROP COLUMN      job_id;

ALTER TABLE dept80
SET UNUSED (last_name);
```

- Lihatlah struktur tabel dept80 dan isi tabelnya, cermati efek dari statement SET UNUSED dan buat catatan.
- Cermati efek dari statement SET UNUSED dan buat catatan pada data di tabel USER_UNUSED_COL_TABS

```
ALTER TABLE dept80
DROP UNUSED COLUMNS;
```

- Dropping a Table
  - Menghapus data dan struktur dari tabel
  - Transaksi yang status *pending* di *commit*
  - Semua index di DROP
  - DROP TABLE tidak bisa dibatalkan

5

```
DROP TABLE dept80;
```

- Merubah nama *Object*

6

```
RENAME dept TO detail_dept;
```

- Truncating a Table
  - Menghapus seluruh data dari sebuah tabel
  - Membebaskan "storage space" yang digunakan oleh tabel tersebut
  - Tidak bisa di Roll Back (alternatif lainnya, gunakan perintah DELETE)

Jalankan satu persatu secara berurutan:

7
```
DELETE FROM c_emps;
SELECT * FROM c_emps;
ROLL BACK;
SELECT * FROM c_emps;
```

8
```
TRUNCATE TABLE detail_dept;
```

- Menambahkan komentar pada tabel & kolom

9
```
COMMENT ON TABLE employees
IS 'Employee Information';

COMMENT ON COLUMN employees.last_name
IS 'Family Name';
```

Cermati isi dari tabel dibawah ini dan buat catatan yang ANDA anggap penting:

- ALL_COL_COMMENTS
- USER_COL_COMMENTS
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

LAB: Creating & Managing Tables

1. Create tabel **my_emp** dengan struktur sbb:

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Data Type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

2. Modifikasi panjang dari last_name menjadi VARCHAR2(50)
3. Pastikan bahwa tabel my_emp sudah masuk dalam dictionary (cek di: USER_TABLES)
4. Isikan tabel my_emp dengan data 4 employee sembarang (Isi dengan nama anda dan 3 teman di sebelah anda)
5. Tambahkan isi tabel my_emp dengan data dari tabel employee yang department_id=50
6. Rename table my_emp menjadi your_emp
7. Pastikan bahwa perubahan nama tabel sudah tercatat dalam dictionary
8. Drop tabel your_emp

Including Constraints

- Memaksa aturan pada level tabel
- Constrains mencegah terjadinya proses delete jika ada pelanggaran aturan
- Type constraints:
  - NOT NULL, mengendalikan bahwa kolom tersebut nilai boleh bernilai null
  - UNIQUE, mengendalikan bahwa kolom atau kombinasi kolom tersebut harus bernilai unik untuk setiap baris dalam tabel (seperti candidate key pada Visual FoxPro)
  - PRIMARY KEY, unik untuk setiap data dalam tabel
  - FOREIGN KEY, untuk menyatakan hubungan antara kolom tersebut dengan kolom pada tabel lainnya.
  - CHECK, mengendalikan sebuah kondisi yang harus selalu ditaati.

- Adding a Constraints

1
```
ALTER TABLE       employees
ADD CONSTRAINT    emp_manager_fk
      FOREIGN KEY(manager_id)
      REFERENCES employees(employee_id);
```

- Dropping a Constraints

2
```
ALTER TABLE  employees
DROP CONSTRAINT emp_manager_fk;
```

---------- jangan dicoba !!! ----------

```
ALTER TABLE  departments
DROP PRIMARY KEY CASCASE;
```

- Disabling Constraints

3
```
ALTER TABLE  employees
DISABLE CONSTRAINT      emp_emp_id_pk CASCADE;
```

- Enabling Constraints

4
```
ALTER TABLE  employees
ENABLE CONSTRAINT       emp_emp_id_pk;
```

- Viewing Constraints

5
```
SELECT    constraint_name, constraint_type, search_condition
FROM      user_constraints
WHERE     table_name = 'EMPLOYEES'
```

6
```
SELECT    constraint_name, column_name
FROM      user_cons_columns
```

7
```
SELECT    * FROM user_objects
```

View

- Digunakan untuk mendapatkan data yang diperlukan, karena view dapat menyajikan data dan kolom yang diperlukan saja
- Membuat query yang kompleks menjadi mudah, karena view dapat menyajikan informasi dari banyak tabel tanpa menuntut user untuk menguasai perintah JOIN
- Menyediakan data independensi untuk ad hoc user dan program aplikasi. Satu view dapat digunakan untuk mengambil data dari beberapa tabel
- Untuk menyajikan data yang sama dalam beberapa sudut pandang

- Classifications View

Feature	Simple Views	Complex Views
Number of tables	One	One or More
Contains functions	No	Yes
Contain groups of data	No	Yes
DML operations through a view	Yes	Not always

- Creating a View

1.
```
CREATE VIEW   empvu80
AS SELECT     employee_id, last_name, salary
     FROM     employees
     WHERE    department_id = 80;
```

2.
```
DESC empvu80;
```

3.
```
CREATE VIEW   salvu50
AS SELECT     employee_id ID_NUMBER, last_name NAME,
              salary*12 ANN_SALARY
     FROM     employees
     WHERE    department_id = 50;
```

4.
```
DESC salvu50;
```

5.
```
SELECT * FROM salvu50;
```

- Modifying a View

```
1. CREATE OR REPLACE VIEW    empvu80
       (id_number, name, sal, department_id)
   AS SELECT    employee_id, first_name  || ' ' || last_name,
                salary, department_id
       FROM     employees
       WHERE    department_id = 80;

2. DESC empvu80;
```

- Creating a Complex View

```
1. CREATE VIEW   dept_sum_vu
       (name, minsal, maxsal, avgsal)
   AS SELECT      d.department_name, MIN(e.salary),
                  MAX(e.salary), AVG(e.salary)
       FROM       employees e, departments d
       WHERE      e.department_id = d.department_id
       GROUP BY   d.department_name;

2. SELECT  *      FROM dept_sum_vu
```

- RULES for Performing DML Operations on a view

ADD data tidak dapat dilakukan jika view mengandung:

- Group functions
- GROUP BY clause
- keyword DISTINCT
- keyword ROWNUM
- Kolom yang dibentuk dengan suatu ekspresi
- Kolom NOT NULL pada tabel dasar yang tidak ikut ditampilkan pada view

- Removing a View

```
DROP VIEW empvu80;
```

- Inline View

```
SELECT     a.last_name, a.salary, a.department_id, b.maxsal
FROM       employees a,  (SELECT     department_id, max(salary) maxsal
                          FROM       employees
                          GROUP BY department_id) b
WHERE      a.department_id = b.department_id
AND        a.salary < b.maxsal;
```

Catatan: Sebuah INLINE VIEW bukanlah sebuah schema object

- Top-*n* Analysis

```
SELECT    ROWNUM as RANK, last_name, salary
FROM      (SELECT last_name, salary FROM employees
           ORDER BY salary DESC)
WHERE     rownum <= 3
```

LAB: View

1. Create view EMPLOYEE_VU yang terdiri dari employee number, employee name, dan department number yang berasal dari tabel EMPLOYEES. Ubah heading dari employee number menjadi EMPLOYEE.

2. Tampilkan isi dari view EMPLOYEE_VU

3. SELECT view_name dan text dari USER_VIEWS di data dictionary view

4. Create view dengan nama DEPT50 yang berisi employee number, employee last names and department number untuk semua employee di department 50. Kolom-kolom view diberi label sbb: EMPNO, EMPLOYEE, DEPNO

5. Tampilkan struktur dan isi dari DEPT50

6. Buatlah query untuk menampilkan 4 karyawan paling senior. (Output: SENIOR, LAST_NAME, HIRE_DATE)

7. Cobalah drop view EMPLOYEE_VU dan DEPT50

Sequence

- Memberikan nomor unik secara otomatis
- Object yang dapat dishare
- Cenderung digunakan untuk membentuk sebuah nilai kunci utama *(primary key)*

- Creating a Sequence

1.
```
CREATE SEQUENCE  dept_deptid_seq
                 INCREMENT BY 10
                 START WITH 120
                 MAXVALUE 9999
                 NOCACHE
                 NOCYCLE;
```

2.
```
SELECT    sequence_name, min_value, max_value,
          increment_by, last_number
FROM      user_sequences;
```

- Using a Sequence

1. INSERT INTO departments(department_id,
   department_name, location_id)
   VALUES (dept_deptid_seq.NEXTVAL, 'Support', 2500);

2. SELECT dept_deptid_seq.CURRVAL FROM dual,

- Modifying a Sequence

1. ALTER SEQUENCE dept_deptid_seq
   INCREMENT BY 20
   MAXVALUE 999999
   NOCACHE
   NOCYCLE;

2. INSERT INTO departments(department_id,
   department_name, location_id)
   VALUES (dept_deptid_seq.NEXTVAL, 'COMP', 2500);

- Removing a Sequence

```
DROP SEQUENCE dept_deptid_seq;
```

LAB: Sequence

1. Create sebuah sequence yang akan digunakan pada kolom primary key di tabel DEPT. Start 200, Max 1000, Increment 10. Beri nama: DEPT_ID SEQ

2. Tuliskan SQL statement untuk melihat sequence_name, min_value, max_value, increment_by dan last_number.

3. Tuliskan SQL statement untuk memasukkan 2 record ke dalam tabel DEPT dengan menggunakan sequence DEPT_ID SEQ untuk mengisikan department_id. (2 Department yang diinput adalah: Education dan Administration)

4. Pastikan kedua data sudah masuk dalam tabel DEPT

5. Berapakah nilai department_id berikutnya?

6. Hapus sequence DEPT_ID SEQ

Index

- Merupakan sebuah schema object
- Digunakan dan dirawat secara otomatis oleh Oracle Server

- Creating an index

1.
```
CREATE INDEX   emp_last_name_idx
ON             employees(last_name);
```

2.
```
SELECT    ic.index_name, ic.column_name,
          ic.column_position col_pos, ix.uniqueness
FROM      user_indexes ix, user_ind_columns ic
WHERE     ic.index_name = ix.index_name
AND       ic.table_name = 'EMPLOYEES';
```

- Function-Based Indexes

1.
```
CREATE INDEX   upper_dept_name_idx
ON             departments(UPPER(department_name));
```

2.
```
SELECT    *
FROM      departments
WHERE     UPPER(department_name) = 'SALES';
```

- Removing an Indexes

```
DROP INDEX    upper_dept_name_idx;

DROP INDEX    emp_last_name_idx;
```

Synonyms

- Mempermudah untuk mengingat tabel yang dimiliki oleh user lain
- Menyingkat nama object yang panjang

- Creating and Removing Synonyms

1.
```
CREATE SYNONYM d_sum
FOR    dept_sum_vu
```

2.
```
DROP SYNONYM d_sum
```

PL/SQL BLOCK STRUCTURE

DECLARE (optional)
variables, cursors, user-defined exceptions
BEGIN (Mandatory)
- SQL statements
- PL/SQL statements

EXCEPTION (optional)
Actions to perform when errors occur
END; (Mandatory)

BLOCK TYPES

Anonymous

```
[DECLARE]

BEGIN
   -- statements

[EXCEPTION]

END;
/
```

Procedure

```
PROCEDURE name
IS

BEGIN
   -- statements

[EXCEPTION]

END;
/
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
   -- statements
   RETURN value;
[EXCEPTION]

END;
/
```

EXAMPLE 1:

```
VARIABLE g_salary NUMBER
BEGIN
        SELECT      salary
        INTO        :g_salary
        FROM        employees
        WHERE       employee_id=178;
END;
/
PRINT g_salary
```

LAB: Create an anonymous block to output the phrase "My PL/SQL Block Works" to the screen.

EXAMPLE 2:

```
VARIABLE g_char VARCHAR2(30)
VARIABLE g_num  NUMBER

DECLARE
    v_char VARCHAR2(30);
    v_num  NUMBER(11,2);
BEGIN
    v_char := '42 is the answer';
    v_num  := TO_NUMBER(SUBSTR(v_char,1,2));
    :g_char := v_char;
    :g_num  := v_num;
END;
/
PRINT g_char
PRINT g_num
```

EXAMPLE 3:

```
VARIABLE g_monthly_sal NUMBER
DEFINE p_annual_sal=50000

SET VERIFY OFF
DECLARE
    v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
    :g_monthly_sal  := v_sal/12;
END;
/
PRINT g_monthly_sal
```

EXAMPLE 3:

```
SET SERVEROUTPUT ON
DECLARE
    v_flag char(5):= 'Betul';
BEGIN
    v_flag := 'Salah';
    DBMS_OUTPUT.PUT_LINE('Hasil Percobaan :' || v_flag );
END;
```

EXAMPLE 4:

```
SET SERVEROUTPUT OFF
DEFINE p_annual_sal = 60000
DECLARE
    v_sal NUMBER := &p_annual_sal;
BEGIN
    v_sal := v_sal/12;
    DBMS_OUTPUT.PUT_LINE ('The monthly salary is ' || TO_CHAR(v_sal));
END;
```

Untuk membantu ANDA lebih memahami sintaks **SET SERVEROUTPUT** lakukanlah **EXAMPLE 5** dan **EXAMPLE 6**.

Format umum: SET SERVEROUT[PUT] {ON|OFF} [SIZE *n*] [FOR[MAT] {WRA[PPED]|WOR[D_WRAPPED]|TRU[NCATED]}]

```
EXAMPLE 5:  SET SERVEROUTPUT ON FORMAT WORD_WRAPPED
            SET LINESIZE 20
            BEGIN
              DBMS_OUTPUT.PUT_LINE('If there is nothing left to do');
              DBMS_OUTPUT.PUT_LINE('shall we continue with plan B?');
            END;
            /

EXAMPLE 5:  SET SERVEROUTPUT ON FORMAT TRUNCATED
            SET LINESIZE 20
            BEGIN
              DBMS_OUTPUT.PUT_LINE('If there is nothing left to do');
              DBMS_OUTPUT.PUT_LINE('shall we continue with plan B?');
            END;
            /
```

PL/SQL Block Syntax and Guidelines

- *Delimiters:*

Simple Symbols

Symbol	Meaning
+	Addition operator
-	Subtraction/negation operator
*	Multiplication operator
/	Division operator
=	Relational operator
@	Remote access indicator
;	Statement terminator

Compound Symbols

Symbol	Meaning
<>	Relational operator
!=	Relational operator
||	Concatenation operator
--	Single line comment indicator
/*	Beginning comment delimiter
*/	Ending comment delimiter
:=	Assignment operator

- *Identifiers:*
  - Can contain up to 30 characters
  - Must begin with an alphabetic character (**v**_sal)
  - Cannot contain character such as hyphens, slashes, and spaces
  - Should not be reserverd words
- *Literals*
  - Characters and date literals must be enclosed in single quotation marks v_name := ‘Henderson’;
  - Numbers can be simple values or scientific notation
- *Comments*

```
DECLARE
    …..
v_sal NUMBER (9,2)

BEGIN
/* Compute the annual sallary based on the

monthly salary input from the user */

v_sal := :gmonthly_sal * 12;

END;      -- Thisis the end of the block
/         -- runs the PL/SQL block
```

Data Type Conversion

- *TO_CHAR (value, fmt)*
- *TO_DATE (value, fmt)*
- *TO_NUMBER (value, fmt)*

```
v_date := TO_DATE (‘January 13, 2003’, ‘Month DD, YYYY’);

v_date := TO_DATE (’12-JAN- 2003’, ‘DD-MON-YYYY’);
```

NESTED BLOCKS

EXAMPLE 5:

```
<<outer>>
DECLARE
    v_sal     NUMBER(7,2) := 60000;
    v_comm    NUMBER(7,2) := v_sal * 0.20;
    v_message VARCHAR2(255) := 'eligible for commission';

BEGIN
    DECLARE
        v_sal       NUMBER(7,2) := 50000;
        v_comm      NUMBER(7,2) := 0;
        v_total_comp NUMBER(7,2) := v_sal + v_comm;

    BEGIN
        v_message := 'CLERK not' || v_message;
        outer.v_comm := v_sal * 0.30;
    END;
    v_message := 'SALESMAN'||v_message;

END;
```

SQL STATEMENTS IN PL/SQL

- *Retrieving data*

```
SET SERVEROUTPUT ON
DECLARE
    v_sum_sal       NUMBER(10,2);
    v_deptno        NUMBER NOT NULL := 60;  --employees.department_id%TYPE:=60


BEGIN
      SELECT    SUM(salary)    --group function
      INTO      v_sum_sal
      FROM      employees
      WHERE     department_id =v_deptno;
      DBMS_OUTPUT.PUT_LINE ('The sum salary is ' || TO_CHAR(v_sum_sal));
END;
```

- *INSERT*

```
BEGIN
    INSERT INTO employees
    (employee_id, first_name, last_name, email,
    hire_date, job_id, salary)

    VALUES
    (employees_seq.NEXTVAL, 'Ruth', 'Cores', 'RCORES', sysdate, 'AD_ASST', 4000);

END;
```

- *UPDATE*

```
DECLARE
    v_sal_increase   employees.salary%TYPE := 800;

BEGIN
    UPDATE  employees
      SET     salary = salary + v_sal_increase
      WHERE   job_id = 'ST_CLERK';
END;
```

- *DELETE*

```
DECLARE
    v_deptno employees.department_id%TYPE := 10;

BEGIN
    DELETE FROM employees
    WHERE       department_id=v_deptno;
END;
```

- *MERGING DATA*

```
DECLARE
    v_empno EMPLOYEES.EMPLOYEE_ID%TYPE := 100;

BEGIN
    MERGE INTO copy_emp c
    USING employees e
    ON (e.employee_id = v_empno)

    WHEN MATCHED THEN

        UPDATE SET
                c.first_name     = e.first_name,
                c.last_name      = e.last_name,
                c.email          = e.email,
                c.phone_number   = e.phone_number,
                c.hire_date      = e.hire_date,
                c.job_id         = e.job_id,
                c.salary         = e.salary,
            c.commission_pct = e.commission_pct,
            c.manager_id     = e.manager_id,
            c.department_id  = e.department_id

    WHEN NOT MATCHED THEN
        INSERT VALUES (e.employee_id, e.first_name, e.last_name, e.email,

        e.phone_number, e.hire_date, e.job_id, e.salary, e.commission_pct, e.manager_id,
        e.department_id);

END;
```

NAMING CONVENTIONS

Identifier	Naming Convention	Example
Variable	v_name	v_sal
Constant	c_name	c_company_name
Cursor	name_cursor	emp_cursor
Exception	e_name	e_too_many
Table type	name_table_type	amount_table_type
Table	name_table	countries
Record type	name_record	customer_record
*i*SQL*Plus substitution variable (also refered to as substitution parameter)	p_name	p_sal
*i*SQL*Plus host or bind variable	g_name	g_year_sal

SQL CURSOR

- There are two types of cursors:
  - implicit cursors
  - explicit cursors
- The Oracle server uses implicit cursors to parse and excecute your SQL statements
- Explicit cursors are explicity declared by the programmer
- SQL Cursor attributes:

SQL%ROWCOUNT	Number of rows affected by the moxt recent SQL statement (an integer value)
SQL%FOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement affects one or more rows
SQL%NOTFOUND	Boolean attribute that evaluates to TRUE if the most recent SQL statement does not affect any rows
SQL%ISOPEN	Always evaluates to FALSE because PL/SQL closes implicit cursors immediately after they are executed

EXAMPLE 6:

```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
   v_employee_id employees.employee_id%TYPE := 176;
BEGIN
   DELETE FROM employees
   WHERE       employee_id = v_employee_id;
   :rows_deleted := (SQL%ROWCOUNT || ' row deleted');
END;
```

CASE Expressions

```
SET SERVEROUTPUT ON
DECLARE
  v_grade CHAR(1) := UPPER('&p_grade');
  v_appraisal VARCHAR2(20);
BEGIN
  v_appraisal :=
   CASE v_grade
      WHEN 'A' THEN 'Excellent'
      WHEN 'B' THEN 'Very Good'
      WHEN 'C' THEN 'Good'
      ELSE 'No such grade'
   END;
DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade|| ' Appraisal ' || v_appraisal);
END;
/
```

```
CASE
   WHEN v_grade = 'A' THEN 'Excellent'
   WHEN v_grade = 'B' THEN 'Very Good'
   WHEN v_grade = 'C' THEN 'Good'
   ELSE 'No such grade'
END;
```

Conditional IF statements:

- **IF** – THEN - **END IF**
- **IF** – THEN – ELSE - **END IF**
- **IF** – THEN – ELSIF - **END IF**

Syntax:

```
IF condition THEN
        statements;
[ELSIF condition THEN
        statements;]
[ELSE
        statements;]
END IF;
```

LAB_IF: Ubahlah PL/SQL diatas menggunakan statement IF.

LOGIC TABLES

AND	*True*	*False*	*Null*
True	*True*	*False*	*Null*
False	*False*	*False*	*False*
Null	*Null*	*False*	*Null*

OR	*True*	*False*	*Null*
True	*True*	*True*	*True*
False	*True*	*False*	*Null*
Null	*True*	*Null*	*Null*

NOT	
True	*False*
False	*True*
Null	*Null*

LOOP Statements

Basic LOOP

```
LOOP
   statement1;
   …..
   EXIT [WHEN condition]
END LOOP;
```

WHILE LOOP

```
WHILE condition LOOP
   statement1;
   statement2;
   …..
END LOOP;
```

FOR LOOP

```
FOR counter IN [REVERSE]
      lower_bound..upper_bound LOOP
   statement1;
   statement2;
   …..
END LOOP;
```

LAB_BasicLoop: *Tulis PL/SQL berikut. Sebelum dijalankan, cek dulu isi tabel LOCATIONS supaya dapat memahami perubahan yang terjadi.*

```
DECLARE
  v_country_id   locations.country_id%TYPE := 'CA';
  v_location_id  locations.location_id%TYPE;
  v_counter      NUMBER(2) := 1;
  v_city         locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id) INTO v_location_id FROM locations
  WHERE country_id = v_country_id;
  LOOP
    INSERT INTO locations(location_id, city, country_id)
    VALUES((v_location_id + v_counter), v_city, v_country_id);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 3;
  END LOOP;
END;
/
```

LAB_WHILELoop: *Ubah PL/SQL diatas menggunakan WHILE Loop*
LAB_FORLoop: *Ubah PL/SQL diatas menggunakan FOR Loop*
LAB_ROLLBACK: *Batalkan semua proses penambahan data*

COMPOSITE DATA TYPES

Siapkan tabel berikut

```
CREATE TABLE retired_emps
        (empno     NUMBER(6),
         ename     VARCHAR2(25),
         job       VARCHAR2(10),
         mgr       NUMBER(6),
         hiredate  DATE,
         leavedate DATE,
         sal       NUMBER(8,2),
         comm      NUMBER(2,2),
         deptno    NUMBER(4));
```

```
DEFINE employee_number = 124
DECLARE
  emp_rec    employees%ROWTYPE;
BEGIN
  SELECT * INTO emp_rec
  FROM employees
  WHERE employee_id = &employee_number;
  INSERT INTO retired_emps(empno, ename, job, mgr, hiredate,
                  leavedate, sal, comm, deptno)
  VALUES (emp_rec.employee_id, emp_rec.last_name, emp_rec.job_id,
        emp_rec.manager_id, emp_rec.hire_date, SYSDATE,
        emp_rec.salary, emp_rec.commission_pct,
        emp_rec.department_id);
  COMMIT;
END;
/

SELECT * FROM retired_emps;
```

```
DECLARE
   TYPE emp_rec_type IS RECORD
(employee_id     NUMBER(6),
 first_name      VARCHAR2(20),
 last_name       employees.last_name%TYPE,
 email           VARCHAR2(20),
 phone_number    VARCHAR2(20),
 hire_date       DATE,
 salary          NUMBER(8,2),
 commission_pct  NUMBER(2,2),
 manager_id      NUMBER(6),
department_id    NUMBER(4));
emp_rec          emp_rec_type;
```

Exception Types

- Predefined Oracle Server ---------- implicitly raised
  - NO_DATA_FOUND
  - TOO_MANY_ROWS
  - INVALID_CURSOR
  - VALUE_ERROR
  - DUP_VAL_ON_INDEX
- Nonpredefined Oracle Server ---------- implicitly raised
- User-defined ---------- explicitly raised

Tidak dianggap error oleh ORACLE SERVER, namun ingin dianggap ERROR oleh PL/User

```
DEFINE p_department_desc = 'Information Technology'
DEFINE p_department_number = 300
DECLARE
   e_invalid_department EXCEPTION;
BEGIN
   UPDATE     departments
   SET        department_name = '&department_desc'
   WHERE      department_id = &p_department_number;
   IF SQL%NOTFOUND THEN
     RAISE e_invalid_department;
   END IF;
   COMMIT;
EXCEPTION
   WHEN e_invalid_department THEN
      DBMS_OUTPUT.PUT_LINE ('No such department id.');
END;
/
```

PROCEDURE

1. IN parameter

```
CREATE OR REPLACE PROCEDURE raise_salary
   (p_id IN employees.employee_id%TYPE)
IS
BEGIN
   UPDATE employees
   SET    salary = salary * 1.10
   WHERE  employee_id = p_id;
END raise_salary;
/
```

Untuk menampilkan IN parameter:

```
EXECUTE raise_salary (176)
```

2. OUT parameter

```
CREATE OR REPLACE PROCEDURE query_emp
   (p_id     IN  employees.employee_id%TYPE,
    p_name   OUT employees.last_name%TYPE,
    p_salary OUT employees.salary%TYPE,
    p_comm   OUT employees.commission_pct%TYPE)
IS
BEGIN
   SELECT last_name, salary, commission_pct
   INTO   p_name, p_salary, p_comm
   FROM employees
   WHERE  employee_id = p_id;
END query_emp;
/
```

Untuk menampilkan OUT parameter:

```
VARIABLE g_name   VARCHAR2(25)
VARIABLE g_sal      NUMBER
VARIABLE g_comm  NUMBER

EXECUTE query_emp(171, :g_name, :g_sal, :g_comm)

PRINT g_name;
```

3. IN OUT parameter

```
CREATE OR REPLACE PROCEDURE format_phone
   (p_phone_no IN OUT VARCHAR2)
IS
BEGIN
   p_phone_no := '(' || SUBSTR(p_phone_no,1,3) ||
                 ')' || SUBSTR(p_phone_no,4,3) ||
                 '-' || SUBSTR(p_phone_no,7);
END format_phone;
/
```

Untuk menampilkan IN OUT parameter:

```
VARIABLE g_phone_no VARCHAR2(15)
BEGIN
   :g_phone_no := '8006330575';
END;
/
PRINT g_phone_no
EXECUTE format_phone (:g_phone_no)
PRINT g_phone_no
```

Procedure tersimpan di data dictionary sampai dengan proses DROP
syntax: DROP PROCEDURE *procedure_name*
Lihat procedure yang pernah ditulis & belum di DROP:
syntax: SELECT * FROM user_source
Lihat daftar PROCEDURE/FUNCTION yang ada:
syntax: SELECT * from user_objects
WHERE object_type IN ('PROCEDURE', 'FUNCTION')

FUNCTION

```
CREATE OR REPLACE FUNCTION tax(p_value IN NUMBER)
   RETURN NUMBER IS
BEGIN
   RETURN (p_value * 0.08);
END tax;
/
SELECT employee_id, last_name, salary, tax(salary)
FROM employees
WHERE department_id = 100;
```

CATATAN:

- ELEMENTS at THE DATE FORMAT MODEL

YYYY	Full year in number
YEAR	Year spelled out
MM	Two digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DDD or DD or D	Numeric day of the year, month or week
AM or PM	Meridian indicator
A.M. or P.M.	Meridian indocator with priods
HH or HH12 or HH24	Hour of day, or hour (1-12), or hour (0 – 23)
MI	Minute (0-59)
SS	Second (0-59)
SSSSS	Second past midnight (0-86399)
/ . ,	Punctuation is reproduced in the result
"of the"	Quoted string is reproduced in the result
TH	Ordinal number (example: DDTH for 4TH)
SP	Spelled-out number (example: DDSP for FOUR)
SPTH or THSP	Spelled-out ordinal numbers (example: DDSPYH for FOURTH)

```
SELECT last_name, TO_CHAR(hire_date, 'fmDD Month YYYY') HIREDATE
FROM employees;

SELECT last_name, TO_CHAR(hire_date, 'fmDdspth "of"  Month YYYY fmHH:MI:SS AM') HIREDATE
FROM employees;

SELECT last_name, TO_CHAR(hire_date, 'DD-MON-YYYY') HIREDATE
FROM employees
WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```