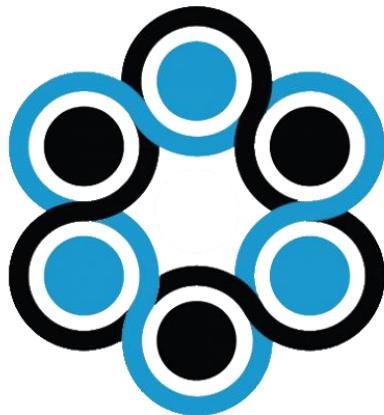


LAPORAN AKHIR
PRAKTIKUM BASIS DATA
Periode V

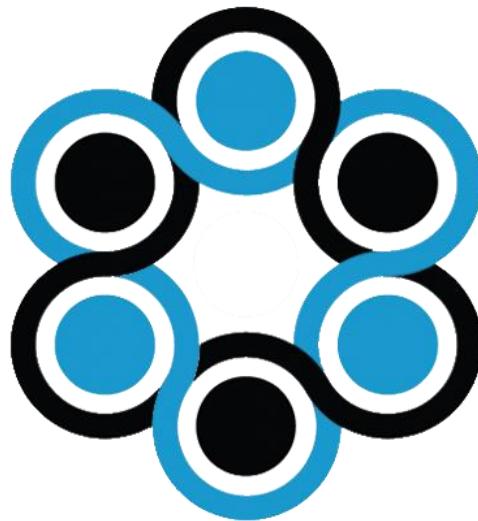


Disusun Oleh :

NAMA : ACHMAD MUCHLASIN
NPM : 06.2018.1.06941

**FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA SURABAYA
SURABAYA
2020**

LAPORAN AKHIR
PRAKTIKUM BASIS DATA
Periode V



Disusun Oleh:

ACHMAD MUCHLASIN 06.2018.1.06941
MARCH ANGGA

**FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA SURABAYA
SURABAYA
2020**

LEMBAR PENGESAHAN

PRAKTIKUM BASIS DATA

Periode V

Di susun oleh :

Achmad Muchlasin

06.2018.1.06941

Tanggal 2/ Juli / 2020

Dosen Pembimbing

Asisten Pembimbing

Septiyawan Rosetya W, S.Kom., M.Kom
NIP 173132

March Angga

NAMA : ACHMAD MUCHLASIN
NPM : 06.2018.1.06941

BIMBINGAN DOSEN
PRAKTIKUM BASIS DATA PERIODE V

No.	TANGGAL	POKOK BAHASAN	PARAF DOSEN PEMBIMBING

**Surabaya,
Dosen Pembimbing**

Septiyawan Rosetya W, S.Kom., M.Kom

NAMA : ACHMAD MUCHLASIN
NPM : 06.2018.1.06941

**BIMBINGAN ASISTEN
PRAKTIKUM BASIS DATA PERIODE V**

No.	TANGGAL	POKOK BAHASAN	PARAF ASISTEN PEMBIMBING

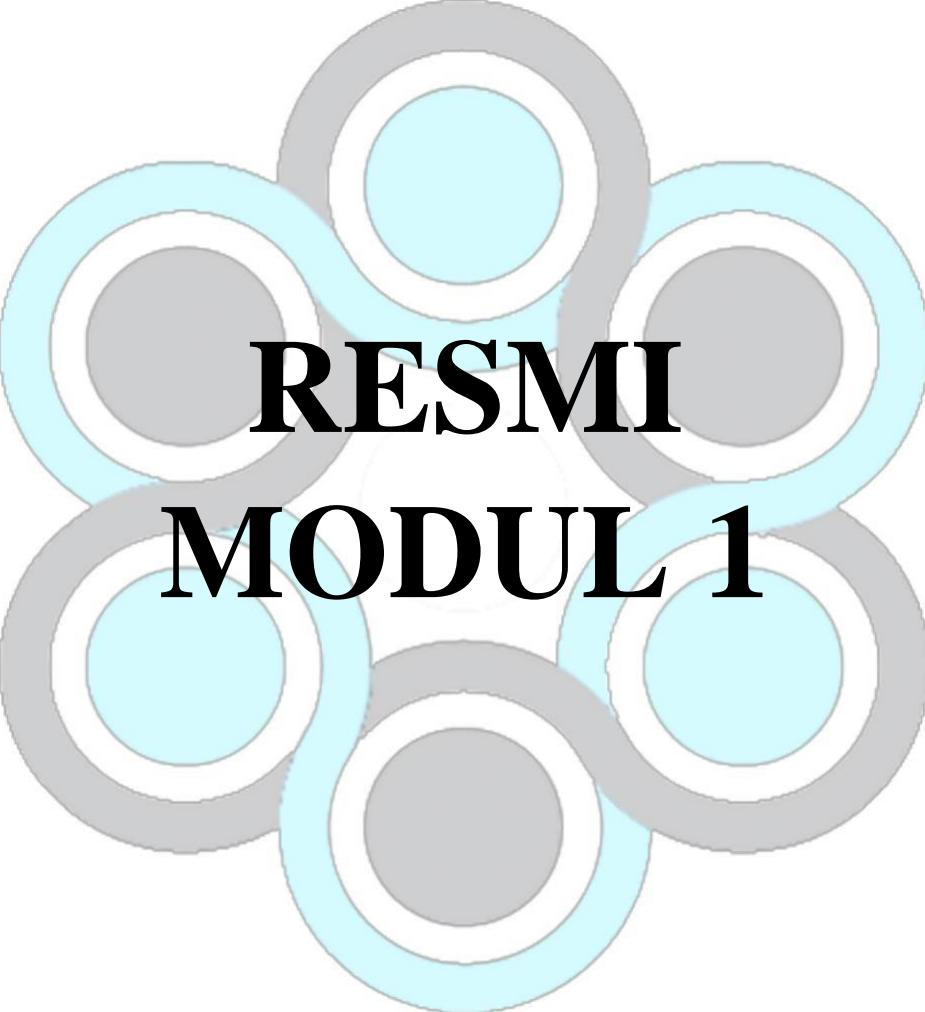
**Surabaya,
Asisten Pembimbing**

March Angga



LAPORAN RESMI

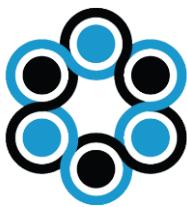
FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



RESMI

MODUL 1

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



Tugas Resmi

Soal Praktikum

1. Buatlah user dan tablespace sesuai Nama dan NPM masing – masing praktikan.
(done)
2. Buatlah tabel sesuai Desain Database Proyek Kelas yang sudah di ACC oleh dosen. (done)
3. Buatlah relasi antar tabel sesuai Desain Database dengan perintah ALTER.
(done)
4. Menerapkan Sequence pada salah satu tabel. (done)
5. Ubah nama salah satu field pada salah satu tabel dengan menambah NPM didepan nama field.
Contoh : 06852_NamaProduk
6. Ubah salah satu field pada salah satu tabel dengan Attribute Constraint UNIQUE.
7. Ubah salah satu field pada salah satu tabel dengan Tipe data yang berbeda.

Langkah Pertama

Membuat tablespace dan user sesuai Nama dan NPM menggunakan salah satu perintah DDL yaitu **CREATE TABLESPACE** dan **CREATE USER**. Login dengan menggunakan user **system**.

Query

```
CREATE TABLESPACE muchlasin_06941
datafile 'C:\Users\Achmad
Muchlasin\praktikum_basdat\outsourcing_comp.d
bf'
size 30M;

CREATE USER muchlasin_06941
IDENTIFIED BY muchlas
DEFAULT TABLESPACE muchlasin_06941
QUOTA 30M ON muchlasin_06941;
```



Tugas Resmi

Screenshot

```
SQL> CREATE TABLESPACE muchlasin_06941
  2  datafile 'C:\Users\Achmad Muchlasin\praktikum_basdat\outsourcing_comp.dbf'
  3  size 30M;

Tablespace created.
```

```
SQL> CREATE USER muchlasin_06941
  2  IDENTIFIED BY muchlas
  3  DEFAULT TABLESPACE muchlasin_06941
  4  QUOTA 30M ON muchlasin_06941;

User created.
```

Analisa

CREATE TABLESPACE

Pada perintah CREATE TABLESPACE terdapat syntax tambahan yaitu:

1) Datafile

Digunakan untuk menentukan lokasi tablespace yang dibuat.

2) Size

Digunakan untuk menentukan ukuran dari tablespace yang telah dibuat.

CREATE USER

Pada perintah CREATE TABLESPACE terdapat syntax tambahan yaitu:

1) Identified by

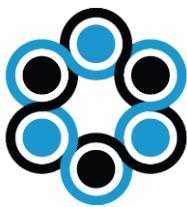
Digunakan untuk meng-set password dari user yang telah dibuat

2) Default tablespace

Digunakan untuk meng-set default tablespace untuk yang yang telah dibuat

3) Quota

Digunakan untuk menentukan kuota yang bisa digunakan oleh user dalam sebuah tablespace.



Tugas Resmi

Langkah Kedua

Setelah membuat TABLESPACE dan USER maka langkah selanjutnya adalah menentukan privileges dari user yang telah kita buat dengan menggunakan perintah **GRANT**. Lalu login dengan user yang telah kita buat tadi dengan perintah **CONN**.

Query

```
GRANT ALL PRIVILEGES TO muchlasin_06941;  
CONN muchlasin_06941/muchlas
```

Screenshot

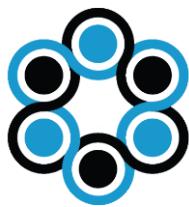
```
SQL> GRANT ALL PRIVILEGES TO muchlasin_06941;  
Grant succeeded.  
  
SQL> CONN muchlasin_06941/muchlas  
Connected.
```

Analisa :

Karena pada langkah paling awal kita masih menggunakan user **system**. Jadi, agar kita bisa membuat privileges yang sama seperti system untuk user baru kita maka kita perlu menggunakan perintah **GRANT ALL PRIVILEGES**. Saat login dengan menggunakan perintah **CONN** maka user baru kita hak aksesnya sudah sama seperti **system** tadi.

Langkah Ketiga

Membuat table sesuai dengan desain yang telah kita buat. Yang pertama kita akan membuat table **Employee**. Dengan beberapa kolom yaitu, **id_employee**, **employee_name**, **employee_addr**, dll.



Tugas Resmi

Query

```
CREATE TABLE employee(
    id_employee int not null,
    employee_name VARCHAR2(50),
    employee_addr VARCHAR2(100),
    gender VARCHAR2(5),
    phone_no number(12),
    email number(30),
    CONSTRAINT PK_employee PRIMARY KEY
(id_employee)
);  
  
DESC employee
```

Screenshot

```
SQL> CREATE TABLE employee(
 2      id_employee int not null,
 3      employee_name VARCHAR2(50),
 4      employee_addr VARCHAR2(100),
 5      gender VARCHAR2(5),
 6      phone_no number(12),
 7      email number(30),
 8      CONSTRAINT PK_employee PRIMARY KEY (id_employee)
 9  );  
Table created.
```

```
SQL> desc employee;
Name          Null?    Type
ID_EMPLOYEE      NOT NULL NUMBER(38)
EMPLOYEE_NAME           VARCHAR2(50)
EMPLOYEE_ADDR          VARCHAR2(100)
GENDER                VARCHAR2(5)
PHONE_NO              NUMBER(12)
EMAIL                 NUMBER(30)  
SQL>
```

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Employee berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks



Tugas Resmi

CONSTRAINT untuk menentukan field mana yang merupakan **PRIMARY KEY** (id_employee). Dan field yang dibuat di dalam table Employee adalah :

- **id_employee (PK)** INTEGER
- employee_name VARCHAR2(50)
- employee_addr VARCHAR2(100)
- gender VARCHAR2(5)
- phone_no NUMBER(12)
- email NUMBER(30)

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_employee**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan **PRIMARY KEY**.

Langkah Keempat

Langkah selanjutnya adalah kita akan membuat table **Client**. Dengan beberapa kolom yaitu, id_client, client_name, client_addr, dll.

Query

```
CREATE TABLE client(  
    id_client int not null,  
    client_name VARCHAR2(50),  
    client_addr VARCHAR2(100),  
    client_email varchar(30),  
    client_phone_no number(12),  
    CONSTRAINT PK_client PRIMARY KEY  
    (id_client)  
);  
  
DESC client
```



Tugas Resmi

Screenshot

```
SQL> CREATE TABLE client(
  2      id_client int not null,
  3      client_name VARCHAR2(50),
  4      client_addr VARCHAR2(100),
  5      client_email varchar(30),
  6      client_phone_no number(12),
  7      CONSTRAINT PK_client PRIMARY KEY (id_client)
  8  );

Table created.
```

```
SQL> DESC client
Name          Null?    Type
-----        -----   -----
ID_CLIENT      NOT NULL NUMBER(38)
CLIENT_NAME    VARCHAR2(50)
CLIENT_ADDR    VARCHAR2(100)
CLIENT_EMAIL   VARCHAR2(30)
CLIENT_PHONE_NO NUMBER(12)

SQL>
```

Analisa :

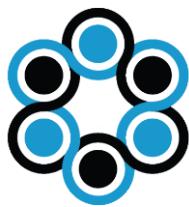
Pada Gambar #1 :

Dapat dilihat query untuk membuat table Client berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks **CONSTRAINT** untuk menentukan field mana yang merupakan **PRIMARY KEY** (`id_client`). Dan field yang dibuat di dalam table Client adalah :

- | | |
|--------------------------------|---------------|
| - id_client (PK) | INTEGER |
| - <code>client_name</code> | VARCHAR2(50) |
| - <code>client_addr</code> | VARCHAR2(100) |
| - <code>client_email</code> | VARCHAR2(30) |
| - <code>client_phone_no</code> | NUMBER(12) |

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_client**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan **PRIMARY KEY**.



Tugas Resmi

Langkah Kelima

Langkah selanjutnya adalah kita akan membuat table **Job**. Dengan beberapa kolom yaitu, id_job, id_client, job_name, dll.

Query

```
CREATE TABLE job(
    id_job int not null,
    id_client int,
    job_name VARCHAR2(100),
    category VARCHAR2(50),
    description VARCHAR2(100),
    CONSTRAINT PK_job PRIMARY KEY (id_job)
);
DESC job
```

Screenshot

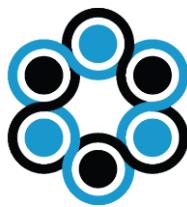
```
SQL> CREATE TABLE job(
  2      id_job int not null,
  3      id_client int,
  4      job_name VARCHAR2(100),
  5      category VARCHAR2(50),
  6      description VARCHAR2(100),
  7      CONSTRAINT PK_job PRIMARY KEY (id_job)
  8  );
```

```
Table created.
```

```
SQL>
```

```
SQL> DESC job
Name          Null?    Type
-----          -----
ID_JOB        NOT NULL NUMBER(38)
ID_CLIENT     NUMBER(38)
JOB_NAME      VARCHAR2(100)
CATEGORY      VARCHAR2(50)
DESCRIPTION   VARCHAR2(100)
```

```
SQL>
```



Tugas Resmi

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Job berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks **CONSTRAINT** untuk menentukan field mana yang merupakan **PRIMARY KEY** (**id_job**). Dan field yang dibuat di dalam table Employee adalah :

- id_job (PK)	INTEGER
- id_client (FK)	INTEGER
- job_name	VARCHAR2(100)
- category	VARCHAR2(50)
- description	VARCHAR2(100)

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_job**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan **PRIMARY KEY**.

Langkah Keenam

Langkah selanjutnya adalah kita akan membuat table **Project**. Dengan beberapa kolom yaitu, **id_project**, **id_job**, **id_employee**, dll.

Query

```
CREATE TABLE project(
    id_project int not null,
    id_job int,
    id_employee int,
    start_date DATE,
    end_date DATE,
    CONSTRAINT PK_project PRIMARY KEY
```



Tugas Resmi

```
);  
DESC project
```

Screenshot

```
SQL> CREATE TABLE project(  
 2      id_project int not null,  
 3      id_job int,  
 4      id_employee int,  
 5      start_date DATE,  
 6      end_date DATE,  
 7      CONSTRAINT PK_project PRIMARY KEY (id_project)  
 8  );  
Table created.
```

```
SQL> DESC project  
Name          Null?    Type  
-----  
ID_PROJECT    NOT NULL NUMBER(38)  
ID_JOB        NUMBER(38)  
ID_EMPLOYEE   NUMBER(38)  
START_DATE    DATE  
END_DATE     DATE  
SQL>
```

Analisa :

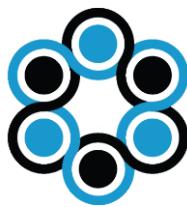
Pada Gambar #1 :

Dapat dilihat query untuk membuat table Client berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks **CONSTRAINT** untuk menentukan field mana yang merupakan **PRIMARY KEY** (`id_project`). Dan field yang dibuat di dalam table Employee adalah :

- **id_employee (PK)** INTEGER
- **id_job (FK)** INTEGER
- **id_employee (FK)** INTEGER
- **start_date** DATE
- **end_date** DATE

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_employee**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan **PRIMARY KEY**.



Tugas Resmi

Langkah Ketujuh

Langkah selanjutnya adalah kita akan membuat relasi antar table dengan menggunakan perintah ALTER.

Query

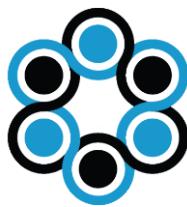
```
ALTER TABLE job
add CONSTRAINT FK_job_client FOREIGN KEY
(id_client)
REFERENCES client (id_client);

ALTER TABLE project
add CONSTRAINT FK_project_employee FOREIGN
KEY (id_employee)
REFERENCES employee(id_employee)
add CONSTRAINT FK_project_job FOREIGN KEY
(id_job)
REFERENCES job(id_job);
```

Screenshot

```
SQL> ALTER TABLE job
2 add CONSTRAINT FK_job_client FOREIGN KEY (id_client)
3 REFERENCES client (id_client);
Table altered.
```

```
SQL> ALTER TABLE project
2 add CONSTRAINT FK_project_employee FOREIGN KEY (id_employee)
3 REFERENCES employee(id_employee)
4 add CONSTRAINT FK_project_job FOREIGN KEY (id_job)
5 REFERENCES job(id_job);
Table altered.
```



Tugas Resmi

Analisa :

Pada Gambar #1 :

Dapat dilihat relasi yang dibuat antara table job dengan client sudah berhasil dengan menggunakan perintah **add CONSTRAINT**. Terdapat juga nama alias **FK_job_client** yang merupakan nama dari **CONSTRAINT FOREIGN KEY** yang kita buat.

Pada Gambar #2 :

Dapat dilihat relasi yang dibuat antara table employee dengan table project dan juga table job dengan table project sudah berhasil dengan menggunakan perintah **add CONSTRAINT**. Fungsi nama alias untuk **CONSTRAINT** yang dibuat sangatlah berguna untuk membedakan **FOREIGN KEY** dari masing-masing table pada database Oracle.

Langkah Kedelapan

Langkah selanjutnya adalah kita membuat sequence yaitu dengan menggunakan perintah **CREATE SEQUENCE**.

Query

```
CREATE SEQUENCE id_project  
MINVALUE 1  
MAXVALUE 100  
START WITH 1  
INCREMENT BY 1  
CACHE 20;
```

Screenshot

```
SQL> CREATE SEQUENCE id_project  
2 MINVALUE 1  
3 MAXVALUE 100  
4 START WITH 1  
5 INCREMENT BY 1  
6 CACHE 20;  
  
Sequence created.
```



Tugas Resmi

Analisa :

Pada gambar di atas, saya membuat sequence untuk table Project dimana field yang menggunakan sequence tersebut adalah **id_project**. Keterangan dari sintaks yang lain juga bisa disimak di bawah ini:

- minvalue : sintaks untuk menentukan nilai terkecil pada sequence.
- maxvalue : sintaks untuk menentukan nilai terakhir pada sequence.
- start with : sintaks untuk menentukan nilai awal pada sequence.
- increment by : sintaks untuk menentukan nilai di tiap pertambahannya.

Langkah Kesembilan

Langkah selanjutnya adalah kita merubah nama field sesuai ketentuan. Perintah yang digunakan yaitu **ALTER – RENAME COLUMN**.

Query

```
ALTER TABLE employee RENAME COLUMN  
id_employee to 06941_id_employee;
```

Screenshot

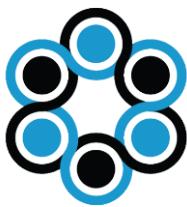
```
SQL> ALTER TABLE employee RENAME COLUMN employee_name TO 06941_employee_name;  
Table altered.
```

Analisa :

Pada gambar di atas nama field sudah berhasil saya ubah sesuai permintaan.

Langkah Kesepuluh

Langkah selanjutnya adalah kita merubah tipe constraint. Pada langkah ini saya menggunakan field yang ada pada table Employee yaitu **phone_no**. Perintah yang digunakan adalah **ALTER - ADD**.



Tugas Resmi

Query

```
ALTER TABLE employee  
add UNIQUE (phone_no);
```

Screenshot

```
SQL> ALTER TABLE employee  
2 add UNIQUE (phone_no);  
Table altered.  
  
SQL> DESC employee  
Name          Null?    Type  
-----  
ID_EMPLOYEE      NOT NULL NUMBER(38)  
06941_EMPLOYEE_NAME  VARCHAR2(50)  
EMPLOYEE_ADDR    VARCHAR2(100)  
GENDER           VARCHAR2(5)  
PHONE_NO         NUMBER(12)  
EMAIL            NUMBER(30)  
  
SQL>
```

Analisa :

Pada gambar di atas menunjukkan bahwa salah satu field yang ada pada table Employee yaitu **phone_no** saya rubah tipe constraint nya dengan menggunakan perintah **ALTER** ditambah dengan **add UNIQUE**. Saya menggunakan field **phone_no** karena tiap orang pasti memiliki no telfon yang berbeda-beda. Perbedaan antara **UNIQUE** dan **PRIMARY KEY** adalah **PRIMARY KEY** selalu bersifat unique namun **UNIQUE** belum tentu **PRIMARY KEY**. Karna constraint **UNIQUE** biasanya digunakan untuk field yang memiliki keunikan di tiap datanya.

Langkah Kesembilan

Langkah selanjutnya adalah kita merubah tipe data dari field dengan menggunakan perintah **ALTER – MODIFY**.

Query

```
ALTER TABLE employee MODIFY(email  
VARCHAR2(30));
```



Tugas Resmi

Screenshot

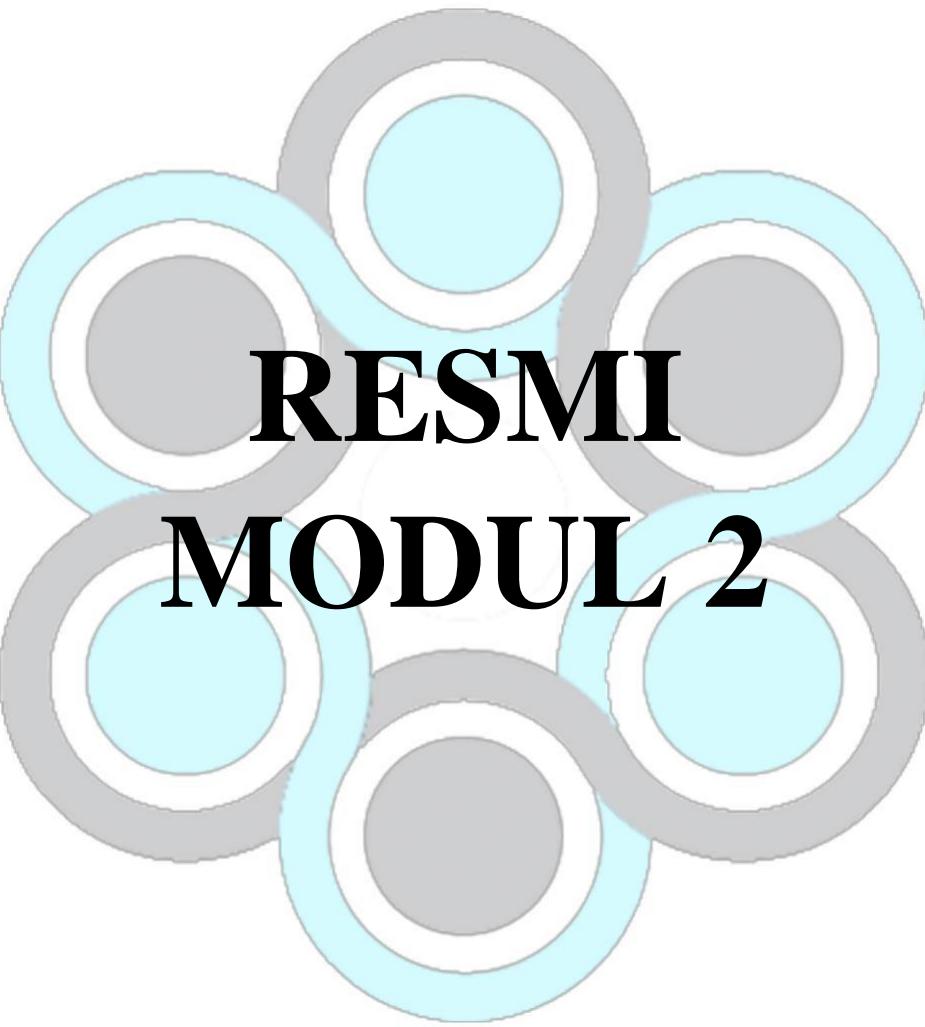
```
SQL> ALTER TABLE employee MODIFY(email VARCHAR2(30));
Table altered.

SQL> DESC employee
Name          Null?    Type
-----        -----
ID_EMPLOYEE   NOT NULL NUMBER(38)
06941_EMPLOYEE_NAME  VARCHAR2(50)
EMPLOYEE_ADDR  VARCHAR2(100)
GENDER         VARCHAR2(5)
PHONE_NO       NUMBER(12)
EMAIL          VARCHAR2(30)

SQL>
```

Analisa :

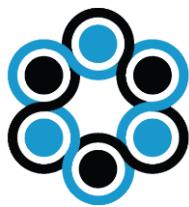
Pada gambar di atas nama salah satu field tipe nya sudah saya ubah menjadi VARCHAR



RESMI

MODUL 2

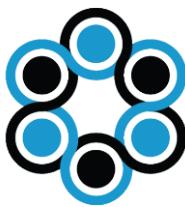
FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



Tugas Resmi

Soal Praktikum

1. Insert data pada setiap tabel yang telah Anda buat :
 - a. Single insert dengan 2 data
 - b. Multiple insert dengan 3 data
 - c. Terapkan insert dengan sequence yang telah Anda buat di Modul 1
2. Update 5 data pada tabel yang berbeda :
 - a. Semua data pada suatu kolom pada tabel
 - b. Menerapkan where clause
 - c. Menerapkan like
 - d. Menerapkan AND, OR dan NOT minimal 2 operator pada satu baris query
3. Delete minimal 3 data pada salah satu tabel dengan menerapkan 1 klausa dan 2 operator pada satu baris query yang berbeda pada setiap data
4. Aturlah transaksi dengan menerapkan :
 - a. Save Point
 - b. Commit
 - c. Rollback
5. Terapkan select dengan menerapkan :
 - a. Order By
 - b. Group By



Tugas Resmi

Langkah Pertama

Insert 5 data pada table **Employee** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO employee (id_employee,
employee_name, employee_addr, gender,
phone_no, email) values (101, 'Muchlas',
'Dusun Duran Sedati', 'L', '8124186',
'muchlas@gmail.com');

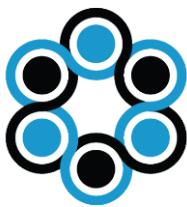
INSERT INTO employee (id_employee,
employee_name, employee_addr, gender,
phone_no, email) values (102, 'Sterling',
'Surabaya Sukolilo', 'L', '8225045',
'sterling@gmail.com');

INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (103, 'Naura', 'Rungkut Surbaya', 'P',
'8774501', 'naura@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (104, 'Alya', 'WR Supratman Surabaya',
'P', '85643801', 'alya@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (105, 'John', 'Gedangan Sidoarjo',
'L', '811345716', 'john@gmail.com')
SELECT 1 FROM dual;)
```

Screenshot

```
SQL> INSERT INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(101, 'Muchlas', 'Dusun Duran Sedati', 'L', '8124186', 'muchlas@gmail.com');
1 row created.

SQL> INSERT INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(102, 'Sterling', 'Surabaya Sukolilo', 'L', '8225045', 'sterling@gmail.com');
1 row created.
```



Tugas Resmi

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (103, 'Naura', 'Rungkut Surbaya', 'P', '8774501', 'naura@gmail.com')
  3  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (104, 'Alya', 'WR Supratman Surabaya', 'P', '85643801', 'alya@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (105, 'John', 'Gedangan Sidoarjo', 'L', '811345716', 'john@gmail.com')
  5  SELECT 1 FROM dual;

3 rows created.

SQL> SELECT * FROM employee;
ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----101 Muchlas
Dusun Duran Sedati
L 8124186 muchlas@gmail.com

102 Sterling
Surabaya Sukolilo
L 8225045 sterling@gmail.com

ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----103 Naura
Rungkut Surbaya
P 8774501 naura@gmail.com

104 Alya
WR Supratman Surabaya

ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----P 85643801 alya@gmail.com

105 John
Gedangan Sidoarjo
L 811345716 john@gmail.com

SQL>
```

Analisa

Gambar #1

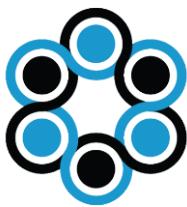
Insert dilakukan pada table **Employee** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INERT INTO** dengan data sebagai berikut:

- 101, 'Muchlas', 'Dusun Duran Sedati', 'L', '8124186', 'muchlas@gmail.com'
- 102, 'Sterling', 'Surabaya Sukolilo', 'L', '8225045', 'sterling@gmail.com'

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.

Gambar #2

Insert dilakukan pada table **Employee** dengan 3 baris selanjutnya menggunakan Multiple Insert menggunakan perintah **INSERT ALL** dengan data sebagai berikut:



Tugas Resmi

- 103, 'Naura', 'Rungkut Surbaya', 'P', '8774501', 'naura@gmail.com'
- 104, 'Alya', 'WR Supratman Surabaya', 'P', '85643801', 'alya@gmail.com'
- 105, 'John', 'Gedangan Sidoarjo', 'L', '811345716', 'john@gmail.com'

Dengan ditambah penggunaan klausa **SELECT 1 FROM dual** pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Employee**. Sesuai query masing – masing (baik single insert maupun multiple insert).

Langkah Kedua

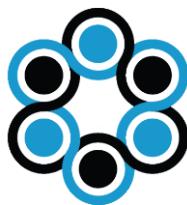
Insert 5 data pada table **Client** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (11, 'Hexamedika', 'Rungkut Surabaya',
'admin@hexa.co.id', '14041');

INSERT INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (12, 'Ekalya Vessel', 'Darmo
Surabaya', 'hrd@ekalya.co.id', '14042');

INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (13, 'Avodamitra', 'Semampir
Sidoarjo', 'hrd@avoda.co.id', '14043')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (14, 'Golden Union', 'Jabon Sidoarjo',
'hr.ga@golden.co.id', '14044')
```



Tugas Resmi

```
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (15, 'Orela Shipyard', 'Pangkah
Gresik', 'support@orela.co.id', '14045')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
1, 'Hexamedika', 'Rungkut Surabaya', 'admin@hexa.co.id', '14041');

1 row created.

SQL> INSERT INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
2, 'Ekalya Vessel', 'Darmo Surabaya', 'hrd@ekalya.co.id', '14042');

1 row created.

SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (13, 'Avo
damitra', 'Semampir Sidoarjo', 'hrd@avoda.co.id', '14043')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (14, 'Gol
den Union', 'Jabon Sidoarjo', 'hr.ga@golden.co.id', '14044')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (15, 'Ore
la Shipyard', 'Pangkah Gresik', 'support@orela.co.id', '14045')
  5  SELECT 1 FROM dual;
```

```
SQL> select * from client;
      ID_CLIENT CLIENT_NAME          CLIENT_ADDR           CLIENT_EMAIL
      CLIENT_PHONE_NO
      -----
      11 Hexamedika          Rungkut Surabaya    admin@hexa.co.id
      14041
      12 Ekalya Vessel        Darmo Surabaya    hrd@ekalya.co.id
      14042
      13 Avodamitra          Semampir Sidoarjo  hrd@avoda.co.id
      14043
      14 Golden Union         Jabon Sidoarjo   hr.ga@golden.co.id
      14044
      15 Orela Shipyard        Pangkah Gresik   support@orela.co.id
      14045
```

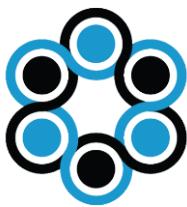
Analisa :

Gambar #1

Insert dilakukan pada table **Client** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- 11, 'Hexamedika', 'Rungkut Surabaya', 'admin@hexa.co.id', '14041'
- 12, 'Ekalya Vessel', 'Darmo Surabaya', 'hrd@ekalya.co.id', '14042'

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.



Tugas Resmi

Gambar #2

Insert dilakukan pada table **Client** dengan 3 baris selanjutnya menggunakan Multiple Insert menggunakan perintah **INSERT ALL** dengan data sebagai berikut:

- 13, 'Avodamitra', 'Semampir Sidoarjo', 'hrd@avoda.co.id', '14043'
- 14, 'Golden Union', 'Jabon Sidoarjo', 'hr.ga@golden.co.id', '14044'
- 15, 'Orela Shipyard', 'Pangkah Gresik', 'support@orela.co.id', '14045'

Dengan ditambah penggunaan klausa SELECT 1 FROM dual pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Client**. Sesuai query masing – masing (baik single insert maupun multiple insert). Tidak ada error saat entry data maupun saat menampilkan data.

Langkah Ketiga

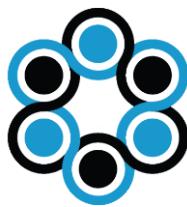
Insert 5 data pada table **Job** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO job (id_job, id_client, job_name,
category, description) values (1101, 11,
'Network Administrator', 'IT', 'Manage and
monitoring network');

INSERT INTO job (id_job, id_client, job_name,
category, description) values (1102, 12,
'Front-End Developer', 'IT', 'Design and
develop UI Website');
```

```
INSERT ALL
INTO job (id_job, id_client, job_name,
category, description) values (1103, 13, 'RAC
Engineer', 'Engineer', 'Install and maintain')
```



Tugas Resmi

```
INTO job (id_job, id_client, job_name,
category, description) values (1104, 14,
'Accounting Staff', 'Finance', 'Reviewing
financial statement')
INTO job (id_job, id_client, job_name,
category, description) values (1105, 15,
'QA/QC Vessel', 'Engineer', 'Inspect and test
materials')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT INTO job (id_job, id_client, job_name, category, description) values (1101, 11, 'Network A
dministrator', 'IT', 'Manage and monitoring network');
1 row created.

SQL> INSERT INTO job (id_job, id_client, job_name, category, description) values (1102, 12, 'Front-End
Developer', 'IT', 'Design and develop UI Website');
1 row created.

SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1103, 13, 'RAC Engineer', '
Engineer', 'Install and maintain cooling system')
  3  INTO job (id_job, id_client, job_name, category, description) values (1104, 14, 'Accounting Staff
', 'Finance', 'Reviewing financial statement')
  4  INTO job (id_job, id_client, job_name, category, description) values (1105, 15, 'QA/QC Vessel', '
Engineer', 'Inspect and test materials')
  5  SELECT 1 FROM dual;
3 rows created.

SQL> select * from job;
   ID_JOB ID_CLIENT JOB_NAME          CATEGORY
DESCRIPTION
----- -----
 1101      11 Network Administrator      IT
           Manage and monitoring network
 1102      12 Front-End Developer       IT
           Design and develop UI Website
 1103      13 RAC Engineer            Engineer
           Install and maintain cooling system
 1104      14 Accounting Staff         Finance
           Reviewing financial statement
 1105      15 QA/QC Vessel            Engineer
           Inspect and test materials
```

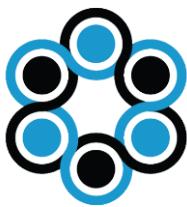
Analisa :

Gambar #1

Insert dilakukan pada table **Job** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- 1101, 11, 'Network Administrator', 'IT', 'Manage and monitoring network'
- 1102, 12, 'Front-End Developer', 'IT', 'Design and develop UI Website'

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.



Tugas Resmi

Gambar #2

Insert dilakukan pada table **Job** dengan 3 baris selanjutnya menggunakan Multiple Insert menggunakan perintah **INSERT ALL** dengan data sebagai berikut:

- 1103, 13, 'RAC Engineer', 'Engineer', 'Install and maintain cooling system'
- 1104, 14, 'Accounting Staff', 'Finance', 'Reviewing financial statement'
- 1105, 15, 'QA/QC Vessel', 'Engineer', 'Inspect and test materials'

Dengan ditambah penggunaan klausa `SELECT 1 FROM dual` pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Job**. Sesuai query masing – masing (baik single insert maupun multiple insert). Tidak ada error saat entry data maupun saat menampilkan data.

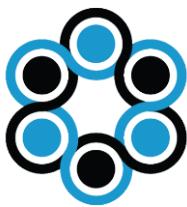
Langkah Keempat

Insert 5 data pada table **Project** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1101, 101,
to_date('01/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy'));

INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1102, 102,
to_date('02/05/2020', 'dd/mm/yyyy'),
to_date('10/05/2020', 'dd/mm/yyyy'));
```



Tugas Resmi

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1103, 103,
to_date('03/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy'))  
  
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1104, 104,
to_date('04/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy'))  
  
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1105, 103,
to_date('05/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy'))
```

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1101, 101, to_date('01/05/2020', 'dd/mm/yyyy'), to_date('20/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1102, 102, to_date('02/05/2020', 'dd/mm/yyyy'), to_date('10/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1103, 103, to_date('03/05/2020', 'dd/mm/yyyy'), to_date('30/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1104, 104, to_date('04/05/2020', 'dd/mm/yyyy'), to_date('20/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1105, 105, to_date('05/05/2020', 'dd/mm/yyyy'), to_date('15/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL>
```

```
SQL> select * from project;  
ID_PROJECT      ID_JOB  ID_EMPLOYEE START_DATE END_DATE  
-----  
      3          1101      101 01-MAY-20 20-MAY-20  
      4          1102      102 02-MAY-20 10-MAY-20  
      6          1103      103 03-MAY-20 30-MAY-20  
      7          1104      104 04-MAY-20 20-MAY-20  
      8          1105      105 05-MAY-20 15-MAY-20
```



Tugas Resmi

Analisa :

Insert dilakukan pada table **Project** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- id_project.nextval, 1101, 101, to_date('01/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy')
- id_project.nextval, 1102, 102, to_date('02/05/2020', 'dd/mm/yyyy'),
to_date('10/05/2020', 'dd/mm/yyyy')

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.

Gambar #2

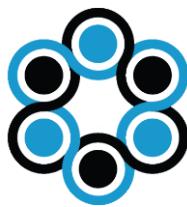
Insert dilakukan pada table **Project** dengan 3 baris selanjutnya masih menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- id_project.nextval, 1103, 103, to_date('03/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy')
- id_project.nextval, 1104, 104, to_date('04/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy')
- id_project.nextval, 1105, 103, to_date('05/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy')

Dengan ditambah penggunaan klausa **SELECT 1 FROM dual** pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Project**. Sesuai query masing – masing (baik single insert maupun multiple insert). Tidak ada error saat entry data maupun saat menampilkan data.



Tugas Resmi

Langkah Kelima

Update data berupa **id_client**, **job_name**, **category** dan **description** pada table **Job** dengan menggunakan query **UPDATE** dasar atau basic.

Query

```
UPDATE job
SET id_client = 15,
job_name = 'QA Vessel',
category = 'Vessel Engineer',
description = 'quality assessment'
WHERE id_job = 1103
```

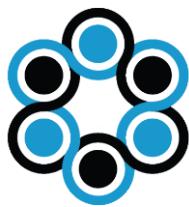
Screenshot

```
SQL> UPDATE job
  2  SET id_client = 15,
  3  job_name = 'QA Vessel',
  4  category = 'Vessel Engineer',
  5  description = 'quality assessment'
  6 WHERE id_job = 1103;
1 row updated.

SQL> SELECT * FROM job;
      ID_JOB ID_CLIENT JOB_NAME          CATEGORY           DESCRIPTION
-----  -----  -----
        1101      11 Network Engineer      IT Support      Maintain and
monitoring network
        1102      12 Network Engineer      IT Support      Maintain, ins
tall, and monitoring network
        1103      15 QA Vessel          Vessel Engineer   quality asses
ment
        1104      14 Accounting Staff     Finance        Reviewing fin
ancial statement
        1105      15 QA/QC Vessel        Engineer       Inspect and t
est materials
SQL>
```

Analisa :

Melakukan update pada table **Job** dengan data yang di update adalah **id_client**, **job_name**, **category** dan **description** dengan kondisi dimana data pada **id_job** memiliki nilai atau berisikan data **1103**. Selain **id_job** yang dientry-kan pada klausa **WHERE** tidak akan mengeksekusi query **UPDATE**.



Tugas Resmi

Langkah Keenam

Update data berupa **id_client**, **job_name**, **category** dan **description** pada table **Client** dengan menggunakan query **UPDATE** dan klausula **WHERE**.

Query

```
UPDATE client
SET client_name = 'Hexa-medika Corp',
client_addr = 'Rungkut Industri Surabaya'
WHERE client_email LIKE 'admin@hexa%' AND
client_name = 'Hexamedika' AND id_client <=
13;
```

Screenshot

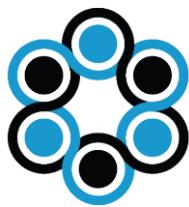
```
SQL> UPDATE client
2 SET client_name = 'Hexa-medika Corp',
3 client_addr = 'Rungkut Industri Surabaya'
4 WHERE client_email LIKE 'admin@hexa%' AND client_name = 'Hexamedika' AND id_client <= 13;
1 row updated.

SQL> SELECT * FROM client;
ID_CLIENT CLIENT_NAME          CLIENT_ADDR           CLIENT_PHONE_NO
-----  -----
11 Hexa-medika Corp      Rungkut Industri Surabaya   14041
12 Ekalya Vessel        Darmo Surabaya            14042
13 CV Avodamitra       Semampir Sidoarjo        14043
14 Golden Union         Jaban Sidoarjo          14044
15 Orela Shipyard       Pangkah Gresik          14045

SQL>
```

Analisa :

Melakukan update pada table **Client** dengan data yang di update adalah **client_name** dan **client_addr** dengan menggunakan klausula **WHERE** untuk kondisi dimana data pada **client_email** memiliki karakteristik kata **admin@hexa** di awal kalimat **DAN** **client_name = Hexamedika DAN id_client <= 13**



Tugas Resmi

Langkah Ketujuh

Update data berupa **employee_name** dan **employee_addr** pada table **Employee** dengan menerapkan klausua **WHERE** dan klausua **LIKE**.

Query

```
UPDATE employee
SET employee_name = 'Naurasari',
employee_addr = 'Taman Bungkul Surabaya'
WHERE email LIKE 'naura%' AND employee_name
LIKE 'Naura%' AND id_employee = 103;
```

Screenshot

```
SQL> UPDATE employee
  2 SET employee_name = 'Naurasari',
  3 employee_addr = 'Taman Bungkul Surabaya'
  4 WHERE email LIKE 'naura%' AND employee_name LIKE 'Naura%' AND id_employee = 103;
1 row updated.

SQL> select * from employee;
ID_EMPLOYEE EMPLOYEE_NAME          EMPLOYEE_ADDR          GEN PHONE_NO EMAIL
-----  -----
          101 Muchlas           Dusun Duran Sedati      L   8124186 muchlas@gmai
mail.com
          102 Sterling          Surabaya Sukolilo       L   8225045 sterling@gmai
gmail.com
          103 Naurasari         Taman Bungkul Surabaya P   8774501 naura@gmai
il.com
          104 Alya              WR Supratman Surabaya P   85643801 alya@gmai
l.com
          105 John              Gedangan Sidoarjo     L   811345716 john@gmai
l.com
SQL>
```

Analisa :

Melakukan update pada table **Employee** dengan data yang di update adalah **employee_name** dan **employee_addr** dengan menerapkan klausua **WHERE** dan klausua **LIKE** untuk kondisi dimana data pada kolom email memiliki karakteristik kata **naura** di awal kalimat **DAN** **employee_name** memiliki karakteristik **Naura** di awal kalimat **DAN** **id_employee = 103**.



Tugas Resmi

Langkah Kedelapan

Update data berupa **client_name** dan **client_email** pada table **Client** dengan menerapkan klausa **WHERE**, klausa **LIKE**, beserta penerapan operator **AND** atau **OR** atau **NOT**.

Query

```
UPDATE client
SET client_name = 'CV Avodamitra',
client_email = 'hrd@avodamitra.co.id'
WHERE client_name LIKE 'Avoda%' OR
client_email LIKE 'hrd@avo%' AND id_client
< 15;
```

Screenshot

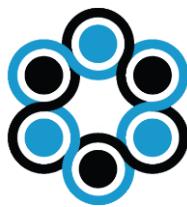
```
SQL> UPDATE client
2 SET client_name = 'CV Avodamitra',
3 client_email = 'hrd@avodamitra.co.id'
4 WHERE client_name LIKE 'Avoda%' OR client_email LIKE 'hrd@avo%' AND id_client < 15;

1 row updated.

SQL> SELECT * FROM client;
ID_CLIENT CLIENT_NAME           CLIENT_ADDR          CLIENT_EMAIL
-----  -----
          11 Hexamedika            Rungkut Surabaya    admin@hexa.co.id
          12 Ekalya Vessel          Darmo Surabaya     hrd@ekalya.co.id
          13 CV Avodamitra          Semampir Sidoarjo   hrd@avodamitra.co.id
          14 Golden Union           Jabon Sidoarjo    hr.ga@golden.co.id
          15 Orela Shipyard          Pangkah Gresik    support@orela.co.id
          14041
          14042
          14043
          14044
          14045
SQL>
```

Analisa :

Melakukan update pada table **Client** dengan data yang di update adalah **client_name** dan **client_email** dengan menerapkan klausa **WHERE** dan klausa **LIKE** untuk kondisi dimana data pada kolom **client_name** memiliki karakteristik kata **Avoda** di awal kalimat **ATAU** **client_email** memiliki karakteristik **hrd@avo%** di awal kalimat **DAN** **id_client < 15**.



Tugas Resmi

Langkah Kesembilan

Update data berupa **start_date** dan **id_employee** pada table **Project** dengan menerapkan klausa **WHERE**, beserta penerapan operator **AND** atau **OR** atau **NOT**.

Query

```
UPDATE project
SET start_date = to_date('07/05/2020',
'dd/mm/yyyy'),
id_employee = 103
WHERE id_project <= 8 AND start_date =
to_date('05/05/2020', 'dd/mm/yyyy') AND
end_date >= to_date('15/05/2020',
'dd/mm/yyyy');
```

Screenshot

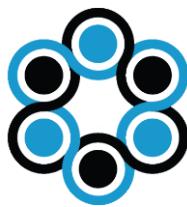
```
SQL> UPDATE project
2 SET start_date = to_date('07/05/2020', 'dd/mm/yyyy'),
3 id_employee = 103
4 WHERE id_project <= 8 AND start_date = to_date('05/05/2020', 'dd/mm/yyyy') AND end_
date >= to_date('15/05/2020', 'dd/mm/yyyy');

1 row updated.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----  -----
      3        1101      101 01-MAY-20 20-MAY-20
      4        1102      102 02-MAY-20 18-MAY-20
      6        1103      103 03-MAY-20 30-MAY-20
      7        1104      104 04-MAY-20 20-MAY-20
      8        1105      103 07-MAY-20 15-MAY-20
```

Analisa :

Melakukan update pada table **Project** dengan data yang di update adalah **start_date** dan **id_employee** dengan menerapkan klausa **WHERE** untuk kondisi dimana data pada kolom **id_project** memiliki kondisi kurang dari sama dengan **8 DAN start_date = 05/05/2020 DAN end_date = 15/05/2020**.



Tugas Resmi

Langkah Kesepuluh

Delete 3 data pada table **Project** dengan penggunaan kondisi yang berbeda – beda.

Query

```
DELETE FROM project
WHERE start_date = to_date('01/05/2020',
'dd/mm/yyyy') AND end_date =
to_date('20/05/2020', 'dd/mm/yyyy') AND
id_project < 8;

DELETE FROM project
WHERE end_date = to_date('20/05/2020',
'dd/mm/yyyy') AND start_date >
to_date('03/05/2020', 'dd/mm/yyyy') OR
id_project = 7;

DELETE FROM project
WHERE start_date BETWEEN
to_date('02/05/2020', 'dd/mm/yyyy') AND
to_date('06/05/2020', 'dd/mm/yyyy') AND
id_project > 4;
```

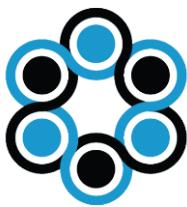
Screenshot

```
SQL> DELETE FROM project
  2 WHERE start_date = to_date('01/05/2020', 'dd/mm/yyyy') AND end_date = to_date('20/05/2020', 'dd/mm/yyyy') AND id_project < 8;
1 row deleted.

SQL> DELETE FROM project
  2 WHERE end_date = to_date('20/05/2020', 'dd/mm/yyyy') AND start_date > to_date('03/05/2020', 'dd/mm/yyyy') OR id_project = 7;
1 row deleted.

SQL> DELETE FROM project
  2 WHERE start_date BETWEEN to_date('02/05/2020', 'dd/mm/yyyy') AND to_date('06/05/2020', 'dd/mm/yyyy') AND id_project > 4;
1 row deleted.

SQL>
```



Tugas Resmi

```
SQL> SELECT * FROM project;  
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE  
-----  
        4          1102      102 02-MAY-20 10-MAY-20  
        8          1105      103 07-MAY-20 15-MAY-20  
SQL>
```

Analisa :

Gambar #1

Melakukan delete dengan menggunakan query **DELETE** dan menerapkan klausa **WHERE** dimana kondisi nya adalah **start_date = 01/05/2020 DAN end_date = 20/05/2020 DAN id_project < 8.**

Gambar #2

Melakukan delete dengan menggunakan query **DELETE** dan menerapkan klausa **WHERE** dimana kondisi nya adalah **end_date = 20/05/2020 DAN start_date lebih dari 03/05/2020 DAN id_project = 7.**

Gambar #3

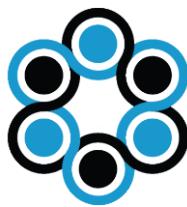
Melakukan delete dengan menggunakan query **DELETE** dan menerapkan klausa **WHERE** dimana kondisi nya adalah **start_date BETWEEN 02/05/2020 DAN 06/05/2020 DAN id_project lebih dari 4.**

Langkah Kesebelas

Melakukan perintah **SAVEPOINT** untuk menyimpan **CHECKPOINT** dimana data terakhir disimpan.

Query

```
SAVEPOINT del_project;
```



Tugas Resmi

Screenshot

```
SQL> SELECT * FROM project;

ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----
        4           1102      102 02-MAY-20 10-MAY-20
        8           1105      103 07-MAY-20 15-MAY-20

SQL> SAVEPOINT del_project;
Savepoint created.

SQL>
```

Analisa :

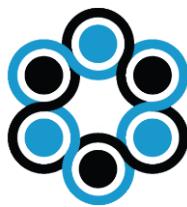
Dapat dilihat pada gambar di atas, dengan menggunakan klaus **SELECT** untuk melihat data yang up to date pada table **Project** untuk selanjutnya dibuatkan **SAVEPOINT** dengan nama **del_project**. Penamaan **SAVEPOINT** sangatlah berpengaruh, untuk memudahkan kita mengingat apabila saat proses entry data ada kesalahan lalu kita ingin **ROLLBACK** ke **SAVEPOINT** yang terakhir kita buat.

Langkah Keduabelas

Melakukan perintah **ROLLBACK** untuk mengembalikan dimana data terakhir disimpan pada pembuatan **SAVEPOINT**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1104, 101,
to_date('10/05/2020', 'dd/mm/yyyy'),
to_date('22/05/2020', 'dd/mm/yyyy'))
ROLLBACK TO SAVEPOINT del_project;
```



Tugas Resmi

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1104, 101, to_date('10/05/2020', 'dd/mm/yyyy'), to_date('22/05/2020
', 'dd/mm/yyyy'));
1 row created.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----      -----      -----      -----      -----
        4          1102        102 02-MAY-20 10-MAY-20
        8          1105        103 07-MAY-20 15-MAY-20
        9          1104        101 10-MAY-20 22-MAY-20

SQL> ROLLBACK TO SAVEPOINT del_project;
Rollback complete.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----      -----      -----      -----      -----
        4          1102        102 02-MAY-20 10-MAY-20
        8          1105        103 07-MAY-20 15-MAY-20

SQL>
```

Analisa :

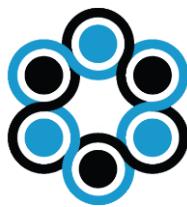
Setelah melakukan **SAVEPOINT** pada langkah sebelumnya, pada langkah ini kita akan mencoba melakukan **ROLLBACK** data dengan percobaan entry data terlebih dahulu pada table **Project**. Setelah entry data pada table **Project**, otomatis data yang ada pada table tersebut akan bertambah. Dengan menggunakan query **ROLLBACK TO SAVEPOINT del_project**, kita bisa mengembalikan kondisi data terakhir saat **SAVEPOINT** dibuat.

Langkah Ketigabelas

Melakukan perintah **COMMIT** untuk menyimpan secara permanen dari perubahan data yang terakhir.

Query

```
COMMIT;
```



Tugas Resmi

Screenshot

```
SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----
        4           1102      102 02-MAY-20 10-MAY-20
        8           1105      103 07-MAY-20 15-MAY-20
SQL> COMMIT;
Commit complete.
SQL>
```

Analisa :

Langkah yang selanjutnya adalah melakukan **COMMIT**. Setelah melakukan **ROLLBACK** pada table **Project** data akan kembali dimana **SAVEPOINT** dibuat. Ketika data tersebut sudah fix dan akan disimpan secara permanen tanpa ada perubahan data lagi, maka kita gunakan query **COMMIT** untuk menyimpan kondisi data terakhir secara permanen (biasanya dilakukan pada akhir entry / update data).

Langkah Keempatbelas

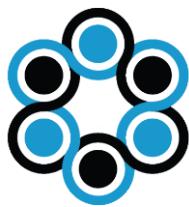
Melakukan perintah **SELECT** dengan menggunakan atau menerapkan kondisi **ORDER BY** pada table **Project**.

Query

```
SELECT * FROM project ORDER BY end_date;
```

Screenshot

```
SQL> SELECT * FROM project ORDER BY end_date;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----
        4           1102      102 02-MAY-20 10-MAY-20
        8           1105      103 07-MAY-20 15-MAY-20
```



Tugas Resmi

Analisa :

Pada langkah ini, bisa dilihat pada gambar di atas dengan menggunakan klausu **SELECT** untuk menampilkan data pada table **Project** ditambah dengan query **ORDER BY** pada kolom `end_date` maka tampilan pada table **Project** akan diurutkan berdasarkan kolom `end_date`.

Langkah Kelimabelas

Melakukan perintah **SELECT** dengan menggunakan atau menerapkan kondisi **GROUP BY** pada table **Project**.

Query

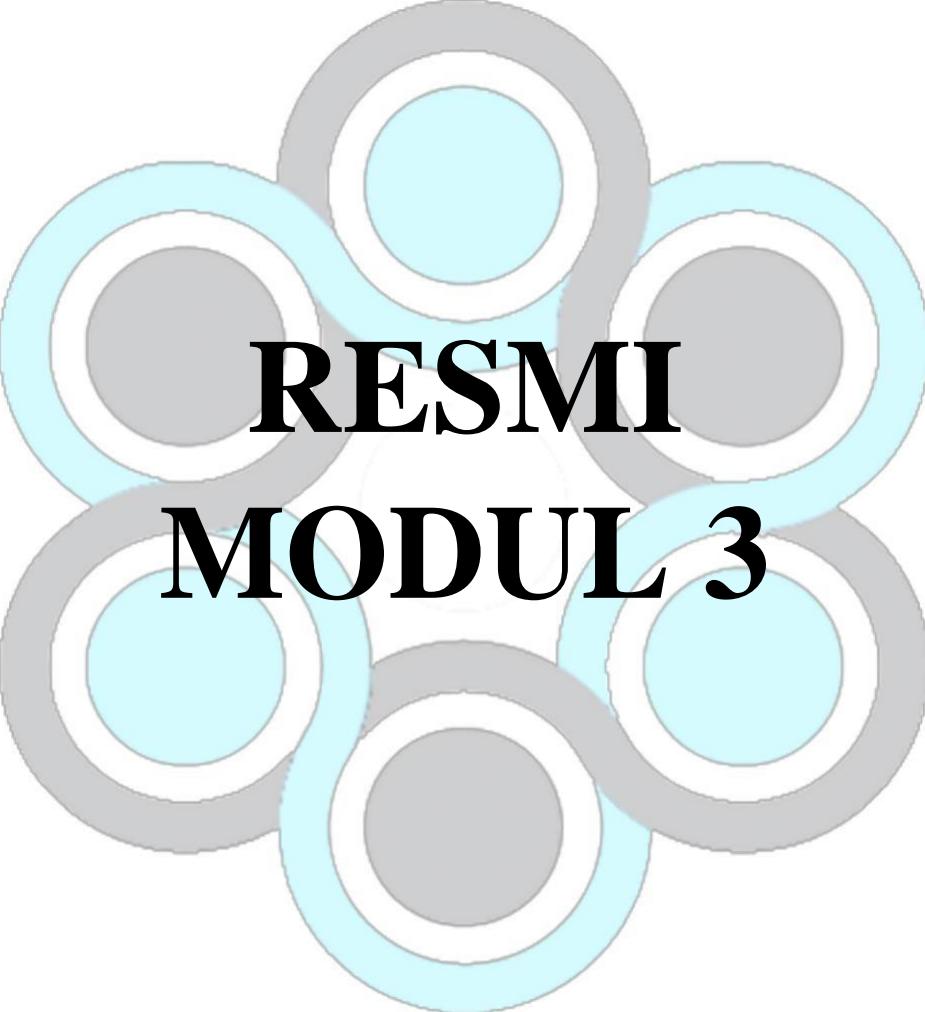
```
SELECT end_date, count(id_job) FROM project  
GROUP BY end_date;
```

Screenshot

```
SQL> SELECT end_date, count(id_job) FROM project GROUP BY end_date;  
END_DATE COUNT(ID_JOB)  
-----  
15-MAY-20      1  
22-MAY-20      1  
SQL>
```

Analisa :

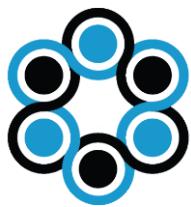
Langkah yang terakhir adalah menampilkan data pada table **Project** yang ditambahkan dengan query **GROUP BY** kolom `end_date`, maka data yang ada pada table **Project** akan menunjukkan pengelompokan data berdasarkan `end_date` dan juga terdapat fungsi agregat **COUNT** untuk menghitung berapa jumlah `id_job` pada tanggal tersebut.



RESMI

MODUL 3

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



Tugas Resmi

Soal Praktikum

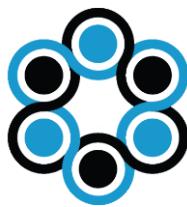
1. Insert minimal 5 data pada setiap tabel

2. Buatlah query dengan menerapkan fungsi aggregate :

- a. Max dan Min dalam 1 Query
- b. Count, 1 klausa dan 2 operator dalam 1 Query
- c. Sum dalam 1 Query
- d. Average pada soal 2C diatas

3. Buatlah Subquery dengan menerapkan :

- a. Max dan Min
- b. Count, 1 klausa dan 2 operator
- c. Sum
- d. Average
- e. Group by
- f. Select setelah where
- g. Minimal 3 nested query



Tugas Resmi

Langkah Pertama

Insert minimal 5 data pada table **Employee** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (106, 'Maya', 'Medaeng', 'P',
'1066666', 'maya@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (107, 'Aga', 'Waru', 'L', '1077777',
'aga@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (108, 'Nara', 'Sidoarjo', 'P',
'1088888', 'nara@gmail.com')
```

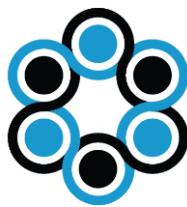
```
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (109, 'Moka', 'Candi', 'L',
'1099999', 'moka@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (110, 'Tara', 'Tanggulangin', 'L',
'1100000', 'tara@gmail.com')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(106, 'Maya', 'Medaeng', 'P', '1066666', 'maya@gmail.com')
  3  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(107, 'Aga', 'Waru', 'L', '1077777', 'aga@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(108, 'Nara', 'Sidoarjo', 'P', '1088888', 'nara@gmail.com')
  5  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(109, 'Moka', 'Candi', 'L', '1099999', 'moka@gmail.com')
  6  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(110, 'Tara', 'Tanggulangin', 'L', '1100000', 'tara@gmail.com')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Tugas Resmi

Analisa

Insert dilakukan pada table **Employee** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Kedua

Insert minimal 5 data pada table **Client** menggunakan **INSERT ALL**.

Query

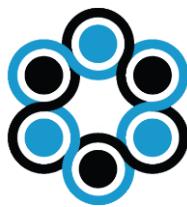
```
INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (16, 'Foxtrot Alto', 'Bukti Darmo',
'hr.ga@foxalto.co', '14444')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (17, 'Padinet', 'Mayjen Sungkono',
'admin.ho@padi.net', '15555')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (18, 'Grand Tower Corp', 'Basuki
Rahmat', 'admin.hr@gtower.co.id', '16666')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (19, 'PT Sejahtera Abadi',
'Tanggulangin Sda', 'inquiry@sa.co.id',
'17777')
```

Screenshot

```
SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  6, 'Foxtrot Alto', 'Bukti Darmo', 'hr.ga@foxalto.co', '14444')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  7, 'Padinet', 'Mayjen Sungkono', 'admin.ho@padi.net', '15555')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  8, 'Grand Tower Corp', 'Basuki Rahmat', 'admin.hr@gtower.co.id', '16666')
  5  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  9, 'PT Sejahtera Abadi', 'Tanggulangin Sda', 'inquiry@sa.co.id', '17777')
  6  INTO client (id_client, client_name, client_addr, client_email, client_email, client_phone_no) values (2
  0, 'Maersk Corp', 'Pandaan', 'hr.ho@maersk.co.id', '18888')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Tugas Resmi

Analisa :

Insert dilakukan pada table **Client** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Ketiga

Insert minimal 5 data pada table **Job** menggunakan **INSERT ALL**.

Query

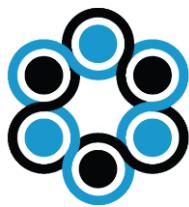
```
INSERT ALL
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (16, 'Foxtrot Alto', 'Bukti Darmo',
    'hr.ga@foxalto.co', '14444')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (17, 'Padinet', 'Mayjen Sungkono',
    'admin.ho@padi.net', '15555')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (18, 'Grand Tower Corp', 'Basuki
    Rahmat', 'admin.hr@gtower.co.id', '16666')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (19, 'PT Sejahtera Abadi',
    'Tanggulangin Sda', 'inquiry@sa.co.id',
    '17777')
```

Screenshot

```
SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1106, 16, 'Software
Engineer', 'IT engineer', 'Maintain website')
  3  INTO job (id_job, id_client, job_name, category, description) values (1107, 18, 'Business
Dev', 'General', 'Collect and submit tender')
  4  INTO job (id_job, id_client, job_name, category, description) values (1108, 16, 'IT Suppor
t', 'IT engineer', 'Maintain network peripheral')
  5  INTO job (id_job, id_client, job_name, category, description) values (1109, 20, 'HRD', 'HR
', 'Manage employee')
  6  INTO job (id_job, id_client, job_name, category, description) values (1110, 20, 'General A
ffair', 'General', 'Manage office tools')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Tugas Resmi

Analisa :

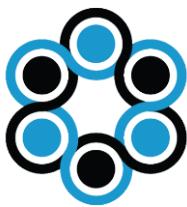
Insert dilakukan pada table **Job** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Keempat

Insert minimal 5 data pada table **Project** menggunakan **INSERT INTO**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1106, 106,
to_date('06/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1107, 107,
to_date('07/05/2020', 'dd/mm/yyyy'),
to_date('18/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1108, 108,
to_date('08/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1109, 107,
to_date('09/05/2020', 'dd/mm/yyyy'),
to_date('21/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1110, 110,
to_date('10/05/2020', 'dd/mm/yyyy'),
to_date('24/05/2020', 'dd/mm/yyyy'));
```



Tugas Resmi

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1106, 106, to_date('06/05/2020', 'dd/mm/yyyy'), to_date('15/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1107, 107, to_date('07/05/2020', 'dd/mm/yyyy'), to_date('18/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1108, 108, to_date('08/05/2020', 'dd/mm/yyyy'), to_date('30/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1109, 107, to_date('09/05/2020', 'dd/mm/yyyy'), to_date('21/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1110, 110, to_date('10/05/2020', 'dd/mm/yyyy'), to_date('14/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL>
```

Analisa :

Insert dilakukan pada table **Project** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Kelima

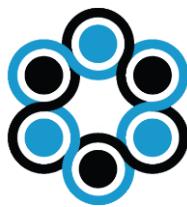
Membuat query dengan menerapkan fungsi aggregate **MIN** dan **MAX** dalam 1 query.

Query

```
SELECT MIN(id_project) AS  
      id_project_terendah, MAX(id_project) as  
      id_project_tertinggi FROM project;
```

Screenshot

```
SQL> select min(id_project) as id_project_terendah,  
 2   max(id_project) as id_project_tertinggi from project;  
  
ID_PROJECT_TERENDAH ID_PROJECT_TERTINGGI  
-----  
2                   25  
  
SQL>
```



Tugas Resmi

Analisa :

Menggunakan fungsi **MIN** untuk menampilkan **id_project_terendah** dan fungsi **MAX** untuk menampilkan **id_project_tertinggi**.

Langkah Keenam

Membuat query dengan menerapkan fungsi aggregate **Count**, 1 klausa dan 2 operator dalam 1 Query.

Query

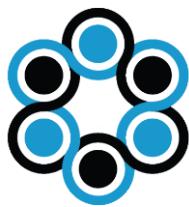
```
select count(id_project) as  
project_dibawah_tgl_8_mei from project  
where start_date < to_date('08/05/2020',  
'dd/mm/yyyy') AND id_project > 2;
```

Screenshot

```
SQL> select count(id_project) as project_dibawah_tgl_8_mei from project  
2 where start_date < to_date('08/05/2020', 'dd/mm/yyyy')  
3 AND id_project > 2;  
  
PROJECT_DIBAWAH_TGL_8_MEI  
-----  
3  
  
SQL>
```

Analisa :

Menggunakan fungsi **COUNT** untuk menghitung jumlah project yang ada pada table **Project** dengan mengambil atau menghitung data berdasarkan **id_project**. Dimana data yang dicari memiliki kondisi tanggal di bawah **8 Mei 2020** dan **id_project** lebih dari 2.



Tugas Resmi

Langkah Ketujuh

Membuat query dengan menerapkan fungsi aggregate **SUM** dalam 1 Query.

Query

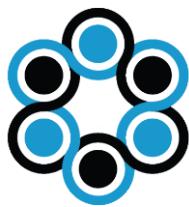
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT SUM(id_client) AS  
2 total_id_client FROM job;  
  
TOTAL_ID_CLIENT  
-----  
157  
  
SQL>
```

Analisa :

Menggunakan fungsi **SUM** untuk menghitung akumulasi dari **id_client** sehingga diperoleh **total_id_client** pada table **Job** dalam 1 line query.



Tugas Resmi

Langkah Kedelapan

Membuat query dengan menerapkan fungsi aggregate **AVERAGE** pada soal 2C diatas

Query

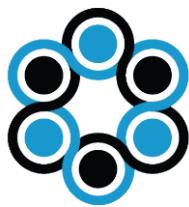
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT AVG(id_client) AS  
2 rerata_total_id_client FROM job;  
  
RERATA_TOTAL_ID_CLIENT  
-----  
15.7  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menghitung rata – rata dari soal nomer 2C yakni jumlah dari **id_client** dan menghasilkan rata – rata dari jumlah tersebut pada table **Job**.



Tugas Resmi

Langkah Kesembilan

Membuat query dengan menerapkan fungsi aggregate **AVERAGE** pada soal 2C diatas

Query

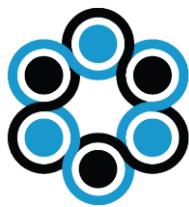
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT AVG(id_client) AS  
2  rerata_total_id_client FROM job;  
  
RERATA_TOTAL_ID_CLIENT  
-----  
          15.7  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menghitung rata – rata dari soal nomer 2C yakni jumlah dari **id_client** dan menghasilkan rata – rata dari jumlah tersebut pada table **Job**.



Tugas Resmi

Langkah Kesepuluh

Membuat subquery dengan menerapkan fungsi aggregate **MAX** dan **MIN**.

Query

```
SELECT id_job, (SELECT MAX(id_client) FROM
job)
AS id_client_max, (SELECT MIN(id_client) FROM
job
AS id_client_min FROM job
WHERE id_client = 20 OR id_client = 11;
```

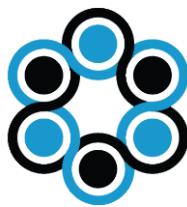
Screenshot

```
SQL> SELECT id_job, (SELECT MAX(id_client) FROM job)
  2 AS id_client_max, (SELECT MIN(id_client) FROM job)
  3 AS id_client_min FROM job
  4 WHERE id_client = 20 OR id_client = 11;

ID_JOB ID_CLIENT_MAX ID_CLIENT_MIN
----- ----- -----
 1109      20          11
 1110      20          11
 1101      20          11
```

Analisa :

Menggunakan fungsi **MAX** dan **MIN** dengan penambahan penggunaan **Subquery** untuk mencari **id_client_max** dan **id_client_min** pada table **Job**. Dimana data yang ditampilkan memiliki kondisi **id_client = 20 ATAU id_client = 11**.



Tugas Resmi

Langkah Kesebelas

Membuat subquery dengan menerapkan fungsi aggregate **Count**, 1 klausula dan 2 operator.

Query

```
SELECT id_client, (SELECT COUNT(id_client)
FROM job
WHERE job.id_client = client.id_client)
AS total_id_client FROM client WHERE
id_client > 10 AND client_name LIKE 'F%' OR
client_name LIKE 'P%';
```

Screenshot

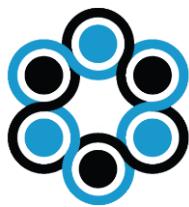
```
SQL> SELECT id_client, (SELECT COUNT(id_client) FROM job
  2 WHERE job.id_client = client.id_client)
  3 AS jumlah_client from client WHERE id_client > 10 AND client_name LIKE 'F%'
  4 OR client_name LIKE 'P%';

ID_CLIENT JUMLAH_CLIENT
-----
16          2
17          0
19          0

SQL>
```

Analisa :

Menggunakan fungsi **COUNT** dengan penambahan penggunaan **Subquery** untuk menampilkan jumlah client pada table **Client**. Dimana data yang ditampilkan memiliki kondisi **id_client** lebih dari **10 DAN client_name** memiliki karakteristik berhuruf depan **F ATAU P**.



Tugas Resmi

Langkah Kedua

Membuat subquery dengan menerapkan fungsi aggregate **SUM**.

Query

```
SELECT id_job, (SELECT SUM(id_job) FROM project  
WHERE project.id_job = job.id_job)  
AS jumlah_id_job FROM job;
```

Screenshot

```
SQL> SELECT id_job, (SELECT SUM(id_job) FROM project  
2 WHERE project.id_job = job.id_job)  
3 AS jumlah_id_job FROM job;  
  
ID_JOB JUMLAH_ID_JOB  
-----  
1101      1101  
1102      1102  
1103  
1104  
1105  
1106      1106  
1107      1107  
1108      1108  
1109      1109  
1110      1110  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **SUM** untuk menghitung akumulasi dari **id_job** pada table **Job** dengan menggunakan alias **jumlah_id_job**.



Tugas Resmi

Langkah Ketigabelas

Membuat subquery dengan menerapkan fungsi aggregate **AVERAGE**

Query

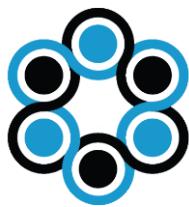
```
SELECT id_client, (SELECT AVG(id_client) FROM job  
WHERE job.id_client = client.id_client)  
AS rerata_id_client FROM client;
```

Screenshot

```
SQL> SELECT id_client, (SELECT AVG(id_client) FROM job  
2 WHERE job.id_client = client.id_client)  
3 AS rerata_id_client FROM client;  
  
ID_CLIENT RERATA_ID_CLIENT  
-----  
11          11  
12          12  
13  
14          14  
15          15  
16          16  
17  
18          18  
19  
20          20  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menampilkan data berupa rata – rata dari akumulasi **id_client** pada table **Client** dengan menggunakan alias **rerata_id_client**.



Tugas Resmi

Langkah Keempatbelas

Membuat subquery dengan menerapkan fungsi **GROUP BY**.

Query

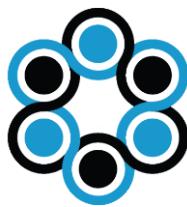
```
SELECT id_employee, (SELECT  
COUNT(id_employee) FROM project  
WHERE project.id_employee =  
employee.id_employee)  
AS jumlah_id_employee FROM employee  
GROUP BY id_employee;
```

Screenshot

```
SQL> SELECT id_employee, (SELECT COUNT(id_employee) FROM project  
2 WHERE project.id_employee = employee.id_employee)  
3 AS jumlah_id_employee FROM employee  
4 GROUP BY id_employee;  
  
ID_EMPLOYEE JUMLAH_ID_EMPLOYEE  
-----  
101          0  
102          2  
103          0  
104          0  
105          0  
106          1  
107          2  
108          1  
109          0  
110          1  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **GROUP BY** untuk menampilkan **id_employee** dan **jumlah_id_employee** pada table **Project**. Dimana data yang ditampilkan memiliki kondisi



Tugas Resmi

Langkah Kelimabelas

Membuat subquery dengan menerapkan fungsi **SELECT** setelah klausu **WHERE**.

Query

```
SELECT id_project, MAX(start_date) AS  
tertinggi, MIN(start_date) AS terendah FROM  
project  
WHERE id_project IN (SELECT id_project FROM  
project WHERE start_date >  
to_date('06/05/2020', 'dd/mm/yyyy'))  
GROUP BY id_project;
```

Screenshot

```
SQL> SELECT id_project, MAX(start_date) AS  
2 tertinggi, MIN(start_date) AS terendah FROM project  
3 WHERE id_project IN (SELECT id_project FROM project WHERE start_date > to_date('06/05/2020  
4 GROUP BY id_project;  
  
ID_PROJECT TERTINGGI TERENDAH  
-----  
      5 07-MAY-20 07-MAY-20  
     22 07-MAY-20 07-MAY-20  
    23 08-MAY-20 08-MAY-20  
    24 09-MAY-20 09-MAY-20  
    25 10-MAY-20 10-MAY-20  
  
SQL>
```

Analisa :

Menggunakan fungsi **SELECT** setelah klausu **WHERE** pada **Subquery** untuk menampilkan **id_project**, **tanggal_tertinggi**, **tanggal_terendah**. Dimana data yang ditampilkan memiliki kondisi **start_date** lebih dari **6 Mei 2020** ditambah dengan penggunaan **GROUP BY** untuk mengelompokkan data berdasarkan **id_project**.



Tugas Resmi

Langkah Keenambelas

Membuat subquery dengan menggunakan minimal 3 nested query.

Query

```
SELECT id_project, (SELECT SUM(id_job) FROM project WHERE id_project < (SELECT COUNT(id_project) AS jumlah_id_project FROM project)) AS akumulasi_id_job FROM project GROUP BY id_project;
```

Screenshot

```
SQL> SELECT id_project, (SELECT SUM(id_job) FROM project WHERE id_project < (SELECT COUNT(id_project) AS jumlah_id_project FROM project)) AS akumulasi_id_job FROM project GROUP BY id_project;
ID_PROJECT AKUMULASI_ID_JOB
-----
      2          2203
      5          2203
     21          2203
     22          2203
     23          2203
     24          2203
     25          2203
7 rows selected.
SQL>
```

Analisa :

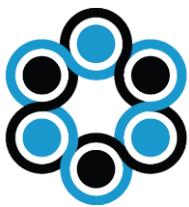
Menggunakan klausua **SELECT** dengan 3 nested query pada pembuatan **Subquery** yang digunakan untuk menampilkan data **id_project** dan **akumulasi_id_job** pada table **Project**. Dimana data yang ditampilkan memiliki kondisi dimana **id_project** harus kurang dari **jumlah_id_project**. Lalu disertai penggunaan **GROUP BY** untuk mengelompokkan data berdasarkan **id_project**.



RESMI

MODUL 4

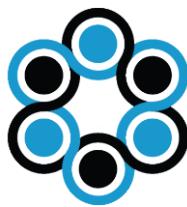
FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



Tugas Resmi

Soal Praktikum

1. Insert 5 data pada setiap tabel
2. Buatlah query laporan dengan jelas sesuai proyek Anda dengan ketentuan sebagai berikut :
 - a. Equi Join minimal 2 tabel dengan menerapkan Operator Pembanding dan Klausa
 - b. Left Join minimal 3 tabel dengan menerapkan Klausa dan Subquery
 - c. Right Join minimal 3 tabel dengan menerapkan Aggregate dan Operator Logika
- Note : Tampilkan kolom pada tabel pertama dan kolom pada tabel kedua dengan memperhatikan kasus yang akan Anda laporkan.
3. Buatlah view dengan menerapkan DML.
4. Buatlah view dari masing-masing soal pada nomor 2.
5. Terapkan DML pada View pada nomor 4.



Tugas Resmi

Langkah Pertama

Insert minimal 5 data pada table **Employee** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (111, 'Roni', 'Sedati', 'L',
'1111111', 'roni@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (112, 'Yuna', 'Waru', 'P', '1222222',
'yuna@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (113, 'Rainy', 'Sidoarjo', 'P',
'1333333', 'rainy@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (114, 'Moza', 'Candi', 'L', '1444444',
'moza@gmail.com')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  3  values (111, 'Roni', 'Sedati', 'L', '1111111', 'roni@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  5  values (112, 'Yuna', 'Waru', 'P', '1222222', 'yuna@gmail.com')
  6  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  7  values (113, 'Rainy', 'Sidoarjo', 'P', '1333333', 'rainy@gmail.com')
  8  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  9  values (114, 'Moza', 'Candi', 'L', '1444444', 'moza@gmail.com')
10  SELECT 1 FROM dual;
11
5 rows created.
```

Analisa

Insert dilakukan pada table **Employee** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Resmi

Langkah Kedua

Insert minimal 5 data pada table **Client** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (21, 'Langitnet', 'Sidoarjo',
'admin@langit.net', '12111')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (22, 'PT PAL', 'Tanjung Perak',
'hrd@pal.co.id', '12222')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (23, 'PT Semen Indonesia', 'Gresik',
'inquiry@semen.co.id', '12333')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (24, 'PT DOK Surabaya', 'Tanjung
Perak', 'admin.hrd@dok.co.id', '12444')
SELECT 1 FROM dual;
```

Screenshot

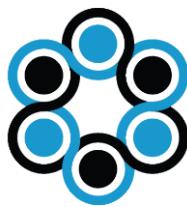
```
SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  6, 'Foxtrot Alto', 'Bukti Darmo', 'hr.g@foxalto.co', '14444')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  7, 'Padinet', 'Mayjen Sungkono', 'admin.ho@padi.net', '15555')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  8, 'Grand Tower Corp', 'Basuki Rahmat', 'admin.hr@gtower.co.id', '16666')
  5  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  9, 'PT Sejahtera Abadi', 'Tanggulangin Sda', 'inquiry@sa.co.id', '17777')
  6  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (2
  0, 'Maersk Corp', 'Pandaan', 'hr.ho@maersk.co.id', '18888')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```

Analisa :

Insert dilakukan pada table **Client** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Resmi

Langkah Ketiga

Insert minimal 5 data pada table **Job** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO job (id_job, id_client, job_name,
category, description) values (1111, 21,
'Admin IT', 'IT', 'Input ticket and generate
ticket report')
INTO job (id_job, id_client, job_name,
category, description) values (1112, 22, 'GA
Drafter', 'Engineer', 'Collect and submit
tender')
INTO job (id_job, id_client, job_name,
category, description) values (1113, 21, 'IT
Support', 'IT', 'Maintain network
peripheral')
INTO job (id_job, id_client, job_name,
category, description) values (1114, 25,
'General Affair', 'General', 'Manage office
tools')
INTO job (id_job, id_client, job_name,
category, description) values (1115, 23,
'Admin HRD', 'HR', 'Input data employee')
SELECT 1 FROM dual;
```

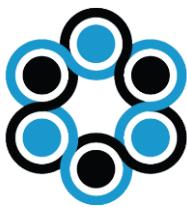
Screenshot

```
SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1111, 21, 'A
dmin IT', 'IT', 'Input ticket and generate ticket report')
  3  INTO job (id_job, id_client, job_name, category, description) values (1112, 22, 'G
A Drafter', 'Engineer', 'Collect and submit tender')
  4  INTO job (id_job, id_client, job_name, category, description) values (1113, 21, 'I
T Support', 'IT', 'Maintain network peripheral')
  5  INTO job (id_job, id_client, job_name, category, description) values (1114, 25, 'G
eneral Affair', 'General', 'Manage office tools')
  6  INTO job (id_job, id_client, job_name, category, description) values (1115, 23, 'A
dmn HRD', 'HR', 'Input data employee')
  7  SELECT 1 FROM dual;

5 rows created.
```

Analisa :

Insert dilakukan pada table **Job** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Resmi

Langkah Keempat

Insert minimal 5 data pada table **Project** menggunakan **INSERT INTO**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1111, 111,
to_date('01/06/2020', 'dd/mm/yyyy'),
to_date('30/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1112, 112,
to_date('02/06/2020', 'dd/mm/yyyy'),
to_date('10/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1113, 113,
to_date('10/06/2020', 'dd/mm/yyyy'),
to_date('30/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1114, 112,
to_date('15/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1111, 111, to_date('01/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1112, 112, to_date('02/06/2020', 'dd/mm/yyyy'), to_date('10/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1113, 113, to_date('10/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

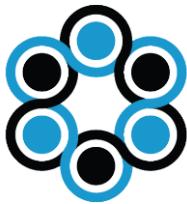
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1114, 112, to_date('15/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1115, 114, to_date('15/06/2020', 'dd/mm/yyyy'), to_date('15/07/2020', 'dd/mm/yyyy'));
1 row created.

SQL>
```

Analisa :

Insert dilakukan pada table **Project** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Resmi

Langkah Kelima

Membuat query dengan menerapkan Equi Join minimal 2 tabel dengan menerapkan Operator Pembanding dan Klausuquery.

Query

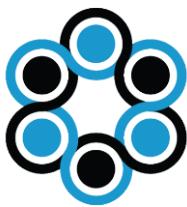
```
select j.id_job, j.job_name, j.description,
c.client_name from job j inner join client c
on j.id_client = c.id_client
where j.id_job <= 1110 order by j.id_job;
```

Screenshot

ID_JOB	JOB_NAME	DESCRIPTION	CLIENT_NAME
1181 Network Administrator	Network Monitoring	Design and monitoring network	Admiral Corp
1182 Front-End Developer	Design and develop UI Website	Design and develop UI Website	Ikalye Vessel
1183 Project Vessel	Design and build of Vessel	Design and build of Vessel	Orca Shipyard
1184 Accounting Staff	Reviewing Financial Statement	Reviewing Financial Statement	Golden Inter
1185 QA/QC Vessel	Inspect and test materials	Inspect and test materials	Orca Shipyard
1186 Quality Control Vessel	Inspect and test materials	Inspect and test materials	Orca Shipyard
1187 Business Dev	Collect and submit tender	Collect and submit tender	Grand Tower Corp
1188 HR Support	Manage employee peripheral	Manage employee peripheral	ICL Corp
1189 HRD	Manage employee	Manage employee	Maersk Corp
1190 General Affairs	Manage office tools	Manage office tools	Maersk Corp

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Equi Join dengan jenis Inner Join. Tabel yang digunakan untuk query di atas adalah Table **Job** dan Table **Client**. Tujuan query di atas adalah menampilkan kolom id_job, job_name, job_description, dan client_name.



Tugas Resmi

Langkah Keenam

Membuat query dengan menerapkan Left Join minimal 3 tabel dengan menerapkan Klausa dan Subquery.

Query

```
select p.id_project, j.job_name,
e.employee_name, p.start_date, p.end_date
from project p left join job j
on p.id_job = j.id_job left join employee e
on p.id_employee = e.id_employee
where p.id_project in (select id_project from
project where start_date <=
to_date('10/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> select p.id_project, j.job_name, e.employee_name, p.start_date, p.end_date from project p left join job j
  2  on p.id_job = j.id_job left join employee e
  3  on p.id_employee = e.id_employee
  4  where p.id_project in (select id_project from project where start_date <= to_date('10/06/2020', 'dd/mm/yyyy'));

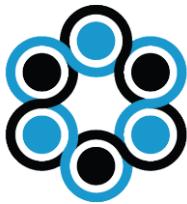
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
-----  -----          -----
21 Software Engineer        Maya             06-MAY-20 15-MAY-20
22 Bussiness Dev           Aga              07-MAY-20 18-MAY-20
23 IT Support               Nara             08-MAY-20 30-MAY-20
24 HRD                      Aga              09-MAY-20 21-MAY-20
25 General Affair          Tara             10-MAY-20 24-MAY-20
41 Admin IT                 Roni             01-JUN-20 30-JUN-20
42 GA Drafter               Yuna             02-JUN-20 10-JUN-20
43 IT Support                Rainy            10-JUN-20 30-JUN-20
2 Front-End Developer       Sterling         02-MAY-20 10-MAY-20
5 Network Administrator     Sterling         07-MAY-20 28-MAY-20

10 rows selected.

SQL>
```

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Left Join dengan menerapkan Klausa dan Subquery pada 3 table yang terlibat. Tabel yang digunakan untuk query di atas adalah **Project, Job, Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, employee_name, start_date, end_date.



Tugas Resmi

Langkah Ketujuh

Membuat query dengan menerapkan Right Join minimal 3 tabel dengan menerapkan Aggregate dan Operator Logika.

Query

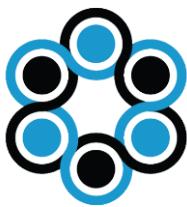
```
select p.id_project, j.job_name,
j.description, e.employee_name, e.email from
project p right join job j
on p.id_job = j.id_job right join employee e
on p.id_employee = e.id_employee
where p.id_project > (select
count(id_project) from project) AND
p.end_date > to_date('30/05/2020',
'dd/mm/yyyy');
```

Screenshot

ID_PROJECT	NAME	DESCRIPTION	EMPLOYEE_NAME	EMAIL
41 Admin IT	Input ticket and generate ticket report		Roni	roni@gmail.com
42 Admin Order	Maintain order peripheral		Roni	roni@gmail.com
43 IT Support	Maintain network peripheral		Rainy	rainy@gmail.com
44 Admin HR	Input data employee		Rainy	rainy@gmail.com
45 Admin HRD			Pola	pola@gmail.com

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Right Join dengan menerapkan Agregate dan Operator Pembanding pada 3 table yang terlibat. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, description, employee_name, email.



Tugas Resmi

Langkah Kedelapan

Membuat view dengan menerapkan DML.

Query

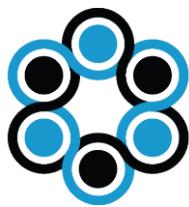
```
CREATE OR REPLACE VIEW CLIENT_EXISTING_V AS  
SELECT id_client, client_name, client_addr,  
client_email FROM client;
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW CLIENT_EXISTING_V AS SELECT id_client, client_name, client_addr, client_email FROM client;  
View created.  
SQL> SELECT * FROM CLIENT_EXISTING_V  
2 ;  
ID_CLIENT CLIENT_NAME CLIENT_ADDR CLIENT_EMAIL  
-----|-----|-----|-----  
16 Footrot Alto Buttil Darroo hr.gm@roxito.co  
17 Padinet Hayjen Sungkono admin.hc@adi.net  
18 PT. Sinar Super Corp Basuki Rahmat amanah.sinar@co.id  
19 PT. Sejahtera Abadi Tenggalingin Sda inquiry@sejaha.co.id  
20 Huersik Corp Pandan hr.huershsk.co.id  
21 IhsanNet Sidoarjo admin.ihnsnet.net  
22 PT Pali Tanjung Perak hr.dgpal.co.id  
23 PT. Sesmen Indonesia Gresik inquiry@sesmen.co.id  
24 PT. SKK Surabaya Tajiung Perak oshikku.skk.co.id  
25 D-Net Basuki Rahmat it@id-net.co.id  
11 Hexamedika Corp Rungkut Industri Surabaya admin.hxhexa.co.id  
  
ID_CLIENT CLIENT_NAME CLIENT_ADDR CLIENT_EMAIL  
-----|-----|-----|-----  
12 Ekalya Vessel Darroo Surabaya hr.gm@ekalya.co.id  
13 CV Avodamitra Maru Sidoarjo hr.gm@avodamitra.co.id  
14 PT. Orela Shipyard Jambatan Pulo hr.gm@orela.co.id  
15 Orela Shipyard Pangkab Gresik support@orela.co.id  
  
IS rows selected.  
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW sederhana untuk memunculkan kolom id_client, client_name, cliend_addr, client_email dari table **Client** dengan nama view **CLIENT_EXISTING_V**. Penamaan dengan sengaja diberi tambahan “V” agar mudah mengetahui bahwa nama tersebut adalah view yang kita buat.



Tugas Resmi

Langkah Kesembilan

Membuat view dari masing-masing soal pada nomor 2a.

Query

```
CREATE OR REPLACE VIEW JOB_FROM_CLIENT_V AS
select j.id_job, j.job_name, j.description,
c.id_client, c.client_name from job j inner
join client c on j.id_client = c.id_client
where j.id_job <= 1110 order by j.id_job;
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW JOB_FROM_CLIENT_V AS select j.id_job, j.job_name, j.description, c.id_client, c.client_na
me from job j inner join client c on j.id_client = c.id_client
2 where j.id_job <= 1110 order by j.id_job;
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Equi Join dengan jenis Inner Join dengan nama **JOB_FROM_CLIENT_V**. Tabel yang digunakan untuk query di atas adalah Table **Job** dan Table **Client**. Tujuan query di atas adalah menampilkan kolom id_job, job_name, job_description, dan client_name.



Tugas Resmi

Langkah Kesepuluh

Membuat view dari masing-masing soal pada nomor 2b.

Query

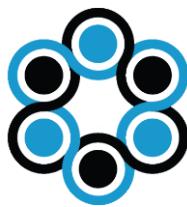
```
CREATE OR REPLACE VIEW PROJECT_UNDER_10JUNE_V
AS
select p.id_project, j.job_name,
e.employee_name, p.start_date, p.end_date
from project p left join job j
on p.id_job = j.id_job left join employee e
on p.id_employee = e.id_employee
where p.id_project in (select id_project from
project where start_date <=
to_date('10/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW PROJECT_UNDER_10JUNE_V AS
  2  select p.id_project, j.job_name, e.employee_name, p.start_date, p.end_date from project p left join j
ob j
  3  on p.id_job = j.id_job left join employee e
  4  on p.id_employee = e.id_employee
  5  where p.id_project in (select id_project from project where start_date <= to_date('10/06/2020', 'dd/m
m/yyyy'));
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Left Join pada 3 table yang terlibat dengan nama **PROJECT_UNDER_10JUNE_V**. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom `id_project`, `job_name`, `employee_name`, `start_date`, `end_date`.



Tugas Resmi

Langkah Kesebelas

Membuat view dari masing-masing soal pada nomor 2c.

Query

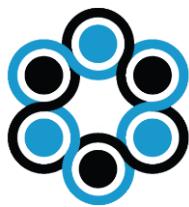
```
CREATE OR REPLACE VIEW DUE_DATE_AFTER_30MAY
AS
select p.id_project, j.job_name,
j.description, e.employee_name, e.email from
project p right join job j
on p.id_job = j.id_job right join employee e
on p.id_employee = e.id_employee
where p.id_project > (select
count(id_project) from project) AND
p.end_date > to_date('30/05/2020',
'dd/mm/yyyy');
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW DUE_DATE_AFTER_30MAY AS
  2  select p.id_project, j.job_name, j.description, e.employee_name, e.email from project p right join jo
  b j
  3  on p.id_job = j.id_job right join employee e
  4  on p.id_employee = e.id_employee
  5  where p.id_project > (select count(id_project) from project) AND p.end_date > to_date('30/05/2020',
dd/mm/yyyy');
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Right Join pada 3 table yang terlibat dengan nama **DUE_DATE_AFTER_30MAY**. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, description, employee_name, email.



Tugas Resmi

Langkah Keduabelas

Menapkan perintah DML pada View nomer 4a.

Query

```
UPDATE JOB_FROM_CLIENT_V
SET job_name = 'NOC'
WHERE id_client = 11;
```

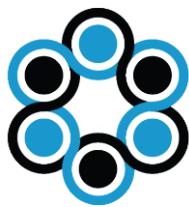
Screenshot

```
SQL> UPDATE JOB_FROM_CLIENT_V
2 SET job_name = 'NOC'
3 WHERE id_client = 11;
1 row updated.

SQL> SELECT * FROM PROJECT_UNDER_10JUNE_V;
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
-----  -----
21 Software Engineer        Maya            06-MAY-20 15-MAY-20
24 HRD                      Aga             09-MAY-20 20-MAY-20
23 Business Dev             Aga             07-MAY-20 18-MAY-20
23 IT Support                Nara            08-MAY-20 30-MAY-20
25 General Affair           Tira            10-MAY-20 24-MAY-20
41 Admin IT                  Roni            01-JUN-20 30-JUN-20
42 GA Drafter               Yuna            02-JUN-20 10-JUN-20
43 IT Support                Rainy           10-JUN-20 30-JUN-20
 2 Front-End Developer       Sterling         02-MAY-20 10-MAY-20
 5 NOC                      Sterling         07-MAY-20 28-MAY-20
10 rows selected.
```

Analisa :

Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **update**. Data yang diupdate disini adalah **job_name** yang awalnya **Network Administrator** menjadi **NOC** dengan mengacu kondisi pada klausa where pada view **JOB_FROM_CLIENT_V**.



Tugas Resmi

Langkah Ketigabelas

Menapkan perintah DML pada View nomer 4b.

Query

```
DELETE FROM PROJECT_UNDER_10JUNE_V
WHERE job_name = 'HRD';
```

Screenshot

```
SQL> DELETE FROM PROJECT_UNDER_10JUNE_V
  2 WHERE job_name = 'HRD';
1 row deleted.

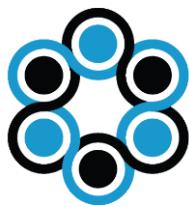
SQL> SELECT * FROM PROJECT_UNDER_10JUNE_V;
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
----- ---------
 21 Software Engineer      Maya
 22 Bussiness Dev          Aga
 23 IT Support              Nara
 25 General Affair         Tara
 41 Admin                  Roni
 42 GA Drafter             Yuna
 43 IT Support              Meliny
  2 Front-End Developer    Sterling
  5 NOC                     Sterling

9 rows selected.

SQL>
```

Analisa :

Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **delete**. Data yang didelete disini adalah `job_name = 'HRD'` dengan mengacu kondisi pada klausula where pada view **PROJECT_UNDER_10JUNE_V**.



Tugas Resmi

Langkah Keempatbelas

Menapkan perintah DML pada View nomer 4c.

Query

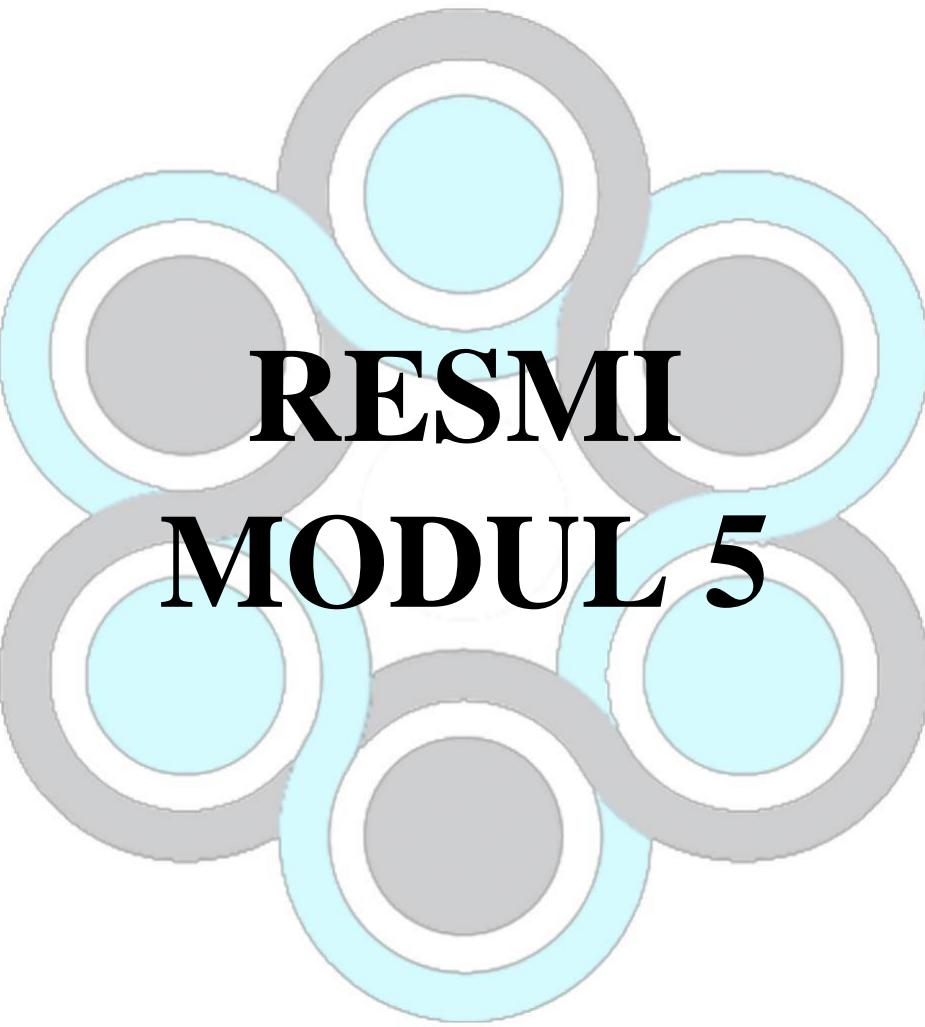
```
SELECT * FROM DUE_DATE_AFTER_30MAY ORDER BY  
id_project DESC;
```

Screenshot

```
SQL> SELECT * FROM DUE_DATE_AFTER_30MAY ORDER BY id_project DESC;  
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      DESCRIPTION           EMAIL  
-----  
45 Admin HRD                Moza              Input data employee    moza@gmail.com  
44 General Affair           Yuna              Manage office tools  yuna@gmail.com  
43 IT Support                Rainy             Maintain network peripheral  rainy@gmail.com  
42 GA Drafter               Yuna              Collect and submit tender  yuna@gmail.com  
41 Admin IT                 Roni              Input ticket and generate ticket report  roni@gmail.com  
SQL>
```

Analisa :

Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **select**. Data yang ditampilkan disini adalah seluruh data dari view yang sudah kita buat yakni **DUE_DATE_AFTER_30MY** dengan tambahan penerapan fungsi **ORDER BY DESC** pada **id_project**.



RESMI

MODUL 5

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



Tugas Resmi

Soal Praktikum

1. Buat koneksi database Oracle dengan Java/Web berdasarkan project masing-masing.
2. Buatlah CRUD pada salah satu tabel (boleh menggunakan GUI, Console atau Web).
3. Terapkan salah satu konsep JOIN dan tampilkan pada project Anda.
4. Terapkan salah satu konsep VIEW dan tampilkan pada project Anda.

Ketentuan :

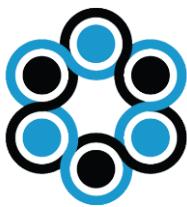
1. Gunakan SQLPlus pada terminal.
2. Kerjakan dalam waktu 150 Menit.

Petunjuk :

1. Bentuk penulisan laporan sementara sesuai template yang ada.
2. Source code dan soal diketik sesuai format yang ada.
3. Upload laporan sementara di classroom dengan batas waktu yang telah ditentukan.
4. Format penamaan upload file :

NPM_NamaPraktikan_LaporanSementara_modul5.

5. Laporan sementara harus diupload pada classroom, jika tidak maka akan terkena pelanggaran dan tidak akan dinilai.



Tugas Resmi

Langkah Pertama

Membuat koneksi database Oracle dengan Java.

Query

```
package com.outsourcingcompany.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class OracleConn {
    private Connection conn;
    private Statement db;
    private String database =
"muchlasin06941";

    public OracleConn() {
        try {

Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Class Driver
Ditemukan");

            try{
                conn =
DriverManager.getConnection("jdbc:oracle:thin
:@localhost:1521:orcl","muchlasin06941"/*user
name*/, "muchlas/*pass*/");
                System.out.println("Koneksi
Database sukses");
            } catch (SQLException se){
                System.out.println("Koneksi
Database gagal = " + se);
            }
        } catch (ClassNotFoundException err){
            System.out.println("Class Driver
Tidak Ditemukan, Terjadi kesalahan pada :
"+err);
        }
    }
}
```



Tugas Resmi

```
        } catch (SQLException se){
            System.out.println("Koneksi
Database gagal = " + se);
        }
    } catch (ClassNotFoundException err){
        System.out.println("Class Driver
Tidak Ditemukan, Terjadi kesalahan pada :
"+err);
    }
}

public ResultSet GetData(String sql){
    try {
        db =
conn.createStatement(ResultSet.TYPE_SCROLL_IN
SENSITIVE, ResultSet.CONCUR_UPDATABLE);
        return db.executeQuery(sql);
    } catch (SQLException e){
        return null;
    }
}

public int ManipulasiData (String sql){
    try {
        db = conn.createStatement();
        return db.executeUpdate(sql);
    } catch (SQLException e){
        return 0;
    }
}
}
```



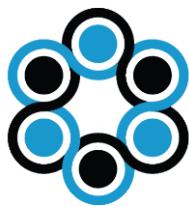
Tugas Resmi

Screenshot

The screenshot shows the 'Output' window from the Netbeans IDE. The title bar reads 'Output - OutsourcingCompany (run)'. The window contains the following text:
run:
Class Driver Ditemukan
Koneksi Database sukses
BUILD SUCCESSFUL (total time: 1 second)

Analisa

Source code di atas adalah source code untuk class **OracleConn.java** yang berfungsi untuk membuat koneksi antara IDE Netbeans dengan Java Oracle agar saling terhubung sesuai dengan database yang diinginkan. Selain menggunakan sintaks tertentu pada class ini, perlu insert driver berupa **odbc7.jar** agar sintaks code yang ditulis bisa dieksekusi sesuai kebutuhan atau keinginan.



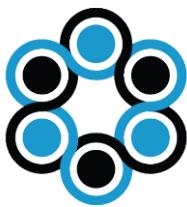
Tugas Resmi

Langkah Kedua

Membuat class **Employee.java** pada Package: **com.outsourcingcompany.model**.

Query

```
public class Employee {  
    private Integer empId;  
    private String empName;  
    private String empAddr;  
    private String empGender;  
    private String empPhoneNo;  
    private String empEmail;  
  
    public Integer getEmpId() {  
        return empId;  
    }  
  
    public void setEmpId(Integer empId) {  
        this.empId = empId;  
    }  
  
    public String getEmpName() {  
        return empName;  
    }  
  
    public void setEmpName(String empName) {  
        this.empName = empName;  
    }  
  
    public String getEmpAddr() {  
        return empAddr;  
    }  
  
    public void setEmpAddr(String empAddr) {  
        this.empAddr = empAddr;  
    }  
  
    public String getEmpGender() {  
        return empGender;  
    }  
  
    public void setEmpGender(String empGender) {  
        this.empGender = empGender;  
    }  
}
```



Tugas Resmi

```
public String getEmpPhoneNo() {
    return empPhoneNo;
}

public void setEmpPhoneNo(String empPhoneNo) {
    this.empPhoneNo = empPhoneNo;
}

public String getEmpEmail() {
    return empEmail;
}

public void setEmpEmail(String empEmail)
{
    this.empEmail = empEmail;
}

}
```

Screenshot

The screenshot shows the NetBeans IDE interface with the Employee.java file open. The code is identical to the one provided above, defining a class Employee with various getters and setters for attributes like empId, empName, empAddr, empGender, empMail, and empPhoneNo.

```
Source | History | File | New | Open | Recent | Favorites | Tools | Window | Help | Exit | 
12 public class Employee {
13     private Integer empId;
14     private String empName;
15     private String empAddr;
16     private String empGender;
17     private String empPhoneNo;
18     private String empEmail;
19
20     public Integer getEmpId() {
21         return empId;
22     }
23
24     public void setEmpId(Integer empId) {
25         this.empId = empId;
26     }
27
28     public String getEmpName() {
29         return empName;
30     }
31
32     public void setEmpName(String empName) {
33         this.empName = empName;
34     }
35
36     public String getEmpAddr() {
37         return empAddr;
38     }
39
40     public void setEmpAddr(String empAddr) {
41         this.empAddr = empAddr;
42     }
43
44     public String getEmpGender() {
45         return empGender;
46     }
47
48     public void setEmpGender(String empGender) {
49         this.empGender = empGender;
50     }
51
52     public String getEmpPhoneNo() {
53         return empPhoneNo;
54     }
55
56     public void setEmpPhoneNo(String empPhoneNo) {
57         this.empPhoneNo = empPhoneNo;
58     }
59
60     public String getEmpEmail() {
```

Analisa :

Source code di atas adalah source code dari **Employee.java** yang berfungsi sebagai model untuk data pegawai yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: empId, empName, empAddr, empGender, empMail, dan empPhoneNo.



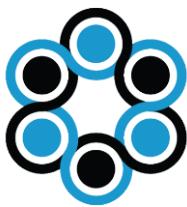
Tugas Resmi

Langkah Keempat

Membuat class **Client.java** pada Package: **com.outsourcingcompany.model**.

Query

```
public class Client {  
    private Integer clId;  
    private String clName;  
    private String clAddr;  
    private String clEmail;  
    private String clPhoneNo;  
  
    public Integer getClId() {  
        return clId;  
    }  
  
    public void setClId(Integer clId) {  
        this.clId = clId;  
    }  
  
    public String getClName() {  
        return clName;  
    }  
  
    public void setClName(String clName) {  
        this.clName = clName;  
    }  
  
    public String getClAddr() {  
        return clAddr;  
    }  
  
    public void setClAddr(String clAddr) {  
        this.clAddr = clAddr;  
    }  
  
    public String getClEmail() {  
        return clEmail;  
    }  
  
    public void setClEmail(String clEmail) {  
        this.clEmail = clEmail;  
    }  
}
```



Tugas Resmi

```
public String getClPhoneNo() {
    return clPhoneNo;
}

public void setClPhoneNo(String
clPhoneNo) {
    this.clPhoneNo = clPhoneNo;
}

}
```

Screenshot

The screenshot shows the NetBeans IDE interface with the code editor open. The code is a Java class named Client. It contains private fields for clId, clName, clAddr, clEmail, and clPhoneNo. It includes methods for getting and setting each of these fields, as well as a constructor that initializes clId to 0. The code is color-coded, with keywords in blue and comments in gray.

```
11  /*
12  */
13  public class Client {
14      private Integer clId;
15      private String clName;
16      private String clAddr;
17      private String clEmail;
18      private String clPhoneNo;
19
20      public Integer getClId() {
21          return clId;
22      }
23
24      public void setClId(Integer clId) {
25          this.clId = clId;
26      }
27
28      public String getClName() {
29          return clName;
30      }
31
32      public void setClName(String clName) {
33          this.clName = clName;
34      }
35
36      public String getClAddr() {
37          return clAddr;
38      }
39
40      public void setClAddr(String clAddr) {
41          this.clAddr = clAddr;
42      }
43
44      public String getClEmail() {
45          return clEmail;
46      }
47
48      public void setClEmail(String clEmail) {
49          this.clEmail = clEmail;
50      }
51
52      public String getClPhoneNo() {
53          return clPhoneNo;
54      }
55
56      public void setClPhoneNo(String clPhoneNo) {
57          this.clPhoneNo = clPhoneNo;
58      }
59  }
```

Analisa :

Source code di atas adalah source code dari **Client.java** yang berfungsi sebagai model untuk data client yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: clId, clName, clAddr, clEmail, clPhoneNo.



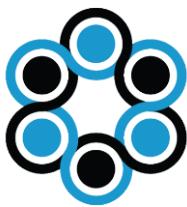
Tugas Resmi

Langkah Kelima

Membuat class **Job.java** pada Package: **com.outsourcingcompany.model**.

Query

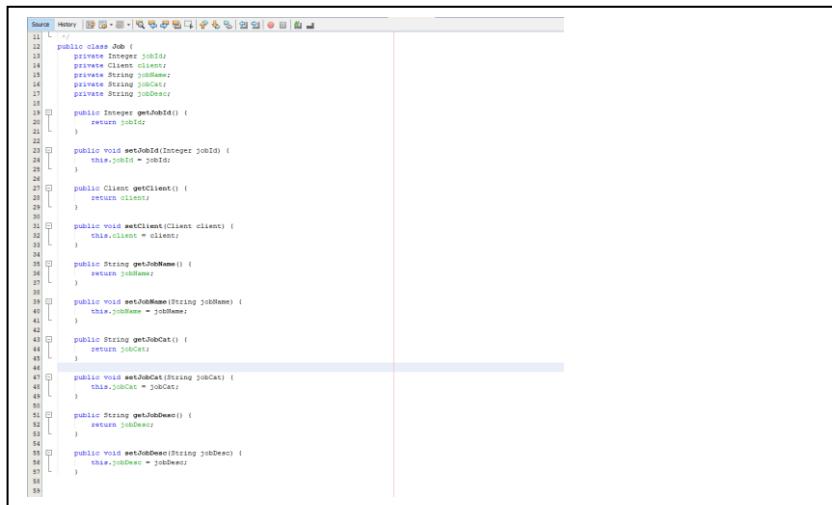
```
public class Job {  
    private Integer jobId;  
    private Client client;  
    private String jobName;  
    private String jobCat;  
    private String jobDesc;  
  
    public Integer getJobId() {  
        return jobId;  
    }  
  
    public void setJobId(Integer jobId) {  
        this.jobId = jobId;  
    }  
  
    public Client getClient() {  
        return client;  
    }  
  
    public void setClient(Client client) {  
        this.client = client;  
    }  
  
    public String getJobName() {  
        return jobName;  
    }  
  
    public void setJobName(String jobName) {  
        this.jobName = jobName;  
    }  
  
    public String getJobCat() {  
        return jobCat;  
    }  
  
    public void setJobCat(String jobCat) {  
        this.jobCat = jobCat;  
    }  
}
```



Tugas Resmi

```
public String getJobDesc() {  
    return jobDesc;  
}  
  
public void setJobDesc(String jobDesc) {  
    this.jobDesc = jobDesc;  
}  
  
}  
}
```

Screenshot



The screenshot shows the NetBeans IDE interface with the source code of the **Job.java** class. The code defines a class with private attributes for jobId, client, jobName, jobCat, and jobDesc, and corresponding getter and setter methods for each.

```
11  /*  
12  *  
13  */  
14  public class Job {  
15      private Integer jobId;  
16      private Client client;  
17      private String jobName;  
18      private String jobCat;  
19      private String jobDesc;  
20      public Integer getJobId() {  
21          return jobId;  
22      }  
23      public void setJobId(Integer jobId) {  
24          this.jobId = jobId;  
25      }  
26      public Client getClient() {  
27          return client;  
28      }  
29      public void setClient(Client client) {  
30          this.client = client;  
31      }  
32      public String getJobName() {  
33          return jobName;  
34      }  
35      public void setJobName(String jobName) {  
36          this.jobName = jobName;  
37      }  
38      public String getJobCat() {  
39          return jobCat;  
40      }  
41      public void setJobCat(String jobCat) {  
42          this.jobCat = jobCat;  
43      }  
44      public String getJobDesc() {  
45          return jobDesc;  
46      }  
47      public void setJobDesc(String jobDesc) {  
48          this.jobDesc = jobDesc;  
49      }  
50  }
```

Analisa :

Source code di atas adalah source code dari **Job.java** yang berfungsi sebagai model untuk data client yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: jobId, client, jobName, jobCat, jobDesc.



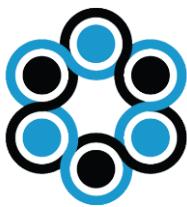
Tugas Resmi

Langkah Keenam

Membuat class **Project.java** pada Package: **com.outsourcingcompany.model**.

Query

```
public class Project {  
    private Integer prjId;  
    private Job job;  
    private Employee employee;  
    private Date startDate;  
    private Date dueDate;  
  
    public Integer getPrjId() {  
        return prjId;  
    }  
  
    public void setPrjId(Integer prjId) {  
        this.prjId = prjId;  
    }  
  
    public Job getJob() {  
        return job;  
    }  
  
    public void setJob(Job job) {  
        this.job = job;  
    }  
  
    public Employee getEmployee() {  
        return employee;  
    }  
  
    public void setEmployee(Employee  
employee) {  
        this.employee = employee;  
    }  
  
    public Date getStartDate() {  
        return startDate;  
    }  
  
    public void setStartDate(Date startDate)  
{  
        this.startDate = startDate;  
    }  
}
```



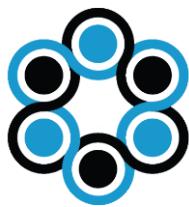
Tugas Resmi

```
public Date getDueDate() {  
    return dueDate;  
}  
  
public void setDueDate(Date dueDate) {  
    this.dueDate = dueDate;  
}  
}
```

Screenshot

Analisa :

Source code di atas adalah source code dari **Project.java** yang berfungsi sebagai model untuk data client yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: prjId, empId, jobId, startDate, dueDate.



Tugas Resmi

Langkah Ketujuh

Membuat class **Controller.java** dengan langkah yang pertama yaitu **import** beberapa **package** serta membuat **constructor**.

Query

```
package com.outsourcingcompany.controller;

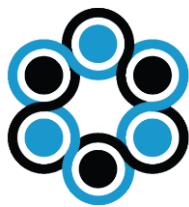
import com.outsourcingcompany.model.Employee;
import com.outsourcingcompany.model.Client;
import com.outsourcingcompany.model.Job;
import com.outsourcingcompany.model.Project;
import
com.outsourcingcompany.database.OracleConn;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class Controller {

    OracleConn conn;

    ArrayList<Employee> arrEmployee;
    ArrayList<Client> arrClient;
    ArrayList<Job> arrJob;
    ArrayList<Project> arrProject;

    public Controller() throws SQLException {
        this.conn = new OracleConn();
        this.arrEmployee = new ArrayList<>();
        this.arrClient = new ArrayList<>();
        this.arrJob = new ArrayList<>();
        this.arrProject = new ArrayList<>();
    }
}
```



Tugas Resmi

Screenshot

```
package com.outsourcingcompany.controller;

import com.outsourcingcompany.model.Employee;
import com.outsourcingcompany.model.Client;
import com.outsourcingcompany.model.Job;
import com.outsourcingcompany.model.Project;
import com.outsourcingcompany.database.OracleConn;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

/**
 * @author Achmad Muchlasin
 */
public class Controller {

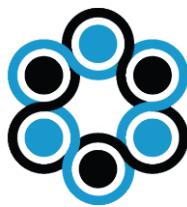
    OracleConn conn;

    ArrayList<Employee> arrEmployee;
    ArrayList<Client> arrClient;
    ArrayList<Job> arrJob;
    ArrayList<Project> arrProject;

    public Controller() throws SQLException {
        this.conn = new OracleConn();
        this.arrEmployee = new ArrayList<>();
        this.arrClient = new ArrayList<>();
        this.arrJob = new ArrayList<>();
        this.arrProject = new ArrayList<>();
    }
}
```

Analisa :

Setelah membuat dan coding di dalam package Model dari **Employee.java**, **Client.java**, **Job.java**, dan **Project.java**. Selanjutnya kita buat class untuk melakukan instruksi transaksi data dari class yang ada di package Model dengan Oracle Database berupa method Create, Read, Update dan Delete. Dapat dilihat di bagian paling atas class ini, perlu import package model, java.sql, dan java.util agar beberapa sintaks code bisa dieksekusi.



Tugas Resmi

Langkah Kedelapan

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getEmployeeData()**.

Query

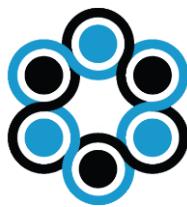
```
public ArrayList<Employee> getEmployeeData() throws  
SQLException {  
    this.arrEmployee.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM EMPLOYEE");  
    while (rs.next()) {  
        Employee emp = new Employee();  
        emp.setEmpId(rs.getInt("ID_EMPLOYEE"));  
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));  
        emp.setEmpAddr(rs.getString("EMPLOYEE_ADDR"));  
        emp.setEmpGender(rs.getString("GENDER"));  
        emp.setEmpPhoneNo(rs.getString("PHONE_NO"));  
        emp.setEmpEmail(rs.getString("EMAIL"));  
        this.arrEmployee.add(emp);  
    }  
    return this.arrEmployee;  
}
```

Screenshot

```
public ArrayList<Employee> getEmployeeData() throws SQLException {  
    this.arrEmployee.clear();  
    ResultSet rs = this.conn.GetData("SELECT * FROM EMPLOYEE");  
    while (rs.next()) {  
        Employee emp = new Employee();  
        emp.setEmpId(rs.getInt("ID_EMPLOYEE"));  
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));  
        emp.setEmpAddr(rs.getString("EMPLOYEE_ADDR"));  
        emp.setEmpGender(rs.getString("GENDER"));  
        emp.setEmpPhoneNo(rs.getString("PHONE_NO"));  
        emp.setEmpEmail(rs.getString("EMAIL"));  
        this.arrEmployee.add(emp);  
    }  
  
    return this.arrEmployee;  
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getEmployeeData()** yang berfungsi untuk membuat mengambil data-data dari **table Employee**.



Tugas Resmi

Langkah Kesembilan

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getClientData()**.

Query

```
public ArrayList<Client> getClientData() throws  
SQLException {  
    this.arrClient.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM CLIENT");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setClId(rs.getInt("ID_CLIENT"));  
        cl.setClName(rs.getString("CLIENT_NAME"));  
        cl.setClAddr(rs.getString("CLIENT_ADDR"));  
        cl.setClEmail(rs.getString("CLIENT_EMAIL"));  
        cl.setClPhoneNo(rs.getString("CLIENT_PHONE_NO"));  
        this.arrClient.add(cl);  
    }  
    return this.arrClient;  
}
```

Screenshot

```
public ArrayList<Client> getClientData() throws SQLException {  
    this.arrClient.clear();  
    ResultSet rs = this.conn.GetData("SELECT * FROM CLIENT");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setClId(rs.getInt("ID_CLIENT"));  
        cl.setClName(rs.getString("CLIENT_NAME"));  
        cl.setClAddr(rs.getString("CLIENT_ADDR"));  
        cl.setClEmail(rs.getString("CLIENT_EMAIL"));  
        cl.setClPhoneNo(rs.getString("CLIENT_PHONE_NO"));  
        this.arrClient.add(cl);  
    }  
  
    return this.arrClient;  
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getClientData()** yang berfungsi untuk membuat mengambil data-data dari **table Client**.



Tugas Resmi

Langkah Kesepuluh

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getJobData()**.

Query

```
public ArrayList<Job> getJobData() throws  
SQLException {  
    this.arrJob.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM JOB JOIN CLIENT ON JOB.ID_CLIENT =  
CLIENT.ID_CLIENT");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setClId(rs.getInt("ID_CLIENT"));  
        cl.setClName(rs.getString("CLIENT_NAME"));  
        cl.setClAddr(rs.getString("CLIENT_ADDR"));  
        cl.setClEmail(rs.getString("CLIENT_EMAIL"));  
  
        cl.setClPhoneNo(rs.getString("CLIENT_PHONE_NO"));  
  
        Job job = new Job();  
        job.setJobId(rs.getInt("ID_JOB"));  
        job.setClient(cl);  
  
        job.setJobName(rs.getString("JOB_NAME"));  
        job.setJobCat(rs.getString("CATEGORY"));  
        job.setJobDesc(rs.getString("DESCRIPTION"));  
        this.arrJob.add(job);  
    }  
  
    return this.arrJob;  
}
```



Tugas Resmi

Screenshot

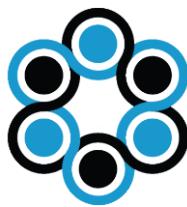
```
public ArrayList<Job> getJobData() throws SQLException {
    this.arrJob.clear();
    ResultSet rs = this.conn.GetData("SELECT * FROM JOB JOIN CLIENT ON JOB.ID_CLIENT = CLIENT.ID_CLIENT");
    while (rs.next()) {
        Client cl = new Client();
        cl.setClientId(rs.getInt("ID_CLIENT"));
        cl.setClnName(rs.getString("CLIENT_NAME"));
        cl.setClAddr(rs.getString("CLIENT_ADDR"));
        cl.setClEmail(rs.getString("CLIENT_EMAIL"));
        cl.setClPhoneNo(rs.getString("CLIENT_PHONE_NO"));

        Job job = new Job();
        job.setJobId(rs.getInt("ID_JOB"));
        job.setClient(cl);
        job.setJobName(rs.getString("JOB_NAME"));
        job.setJobCat(rs.getString("CATEGORY"));
        job.setJobDesc(rs.getString("DESCRIPTION"));
        this.arrJob.add(job);
    }

    return this.arrJob;
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getJobData()** yang berfungsi untuk membuat mengambil data-data dari **table Job**.



Tugas Resmi

Langkah Kesebelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getJobClientData()**.

Query

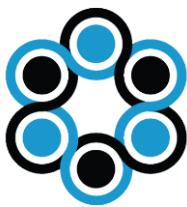
```
public ArrayList<Job> getJobClientData() throws  
SQLException {  
    this.arrJob.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM JOB_FROM_CLIENT_V");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setCId(rs.getInt("ID_CLIENT"));  
        cl.setName(rs.getString("CLIENT_NAME"));  
        Job job = new Job();  
        job.setId(rs.getInt("ID_JOB"));  
        job.setClient(cl);  
        job.setName(rs.getString("JOB_NAME"));  
        job.setDescription(rs.getString("DESCRIPTION"));  
        this.arrJob.add(job);  
    }  
    return this.arrJob;  
}
```

Screenshot

```
public ArrayList<Job> getJobClientData() throws SQLException {  
    this.arrJob.clear();  
    ResultSet rs = this.conn.GetData("SELECT * FROM JOB_FROM_CLIENT_V");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setCId(rs.getInt("ID_CLIENT"));  
        cl.setName(rs.getString("CLIENT_NAME"));  
  
        Job job = new Job();  
        job.setId(rs.getInt("ID_JOB"));  
        job.setClient(cl);  
        job.setName(rs.getString("JOB_NAME"));  
        job.setDescription(rs.getString("DESCRIPTION"));  
        this.arrJob.add(job);  
    }  
  
    return this.arrJob;  
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getJobClientData()** yang berfungsi untuk membuat mengambil data-data dari VIEW yang sudah dibuat antara **table Job** dan **table Client**.



Tugas Resmi

Langkah Keduabelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getProjectData()**.

Query

```
public ArrayList<Project> getProjectData() throws  
SQLException {  
    this.arrProject.clear();  
    ResultSet rs = this.conn.GetData("SELECT  
PROJECT.ID_PROJECT, JOB.JOB_NAME,  
EMPLOYEE.EMPLOYEE_NAME, PROJECT.START_DATE,  
PROJECT.END_DATE FROM PROJECT JOIN JOB ON  
PROJECT.ID_JOB = JOB.ID_JOB JOIN EMPLOYEE ON  
PROJECT.ID_EMPLOYEE = EMPLOYEE.ID_EMPLOYEE ORDER BY  
PROJECT.ID_PROJECT ASC");  
    while (rs.next()) {  
        Employee emp = new Employee();  
  
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));  
  
        Job job = new Job();  
  
        job.setJobName(rs.getString("JOB_NAME"));  
  
        Project project = new Project();  
  
        project.setPrjId(rs.getInt("ID_PROJECT"));  
        project.setJob(job);  
        project.setEmployee(emp);  
  
        project.setStartDate(rs.getDate("START_DATE"));  
  
        project.setDueDate(rs.getDate("END_DATE"));  
  
        this.arrProject.add(project);  
    }  
  
    return this.arrProject;  
}
```



Tugas Resmi

Screenshot

```
public ArrayList<Project> getProjectData() throws SQLException {
    this.arrProject.clear();
    ResultSet rs = this.conn.GetData("SELECT PROJECT.ID_PROJECT, "
        + "JOB.JOB_NAME, EMPLOYEE.EMPLOYEE_NAME, PROJECT.START_DATE, "
        + "PROJECT.END_DATE FROM PROJECT JOIN JOB "
        + "ON PROJECT.ID_JOB = JOB.ID_JOB "
        + "JOIN EMPLOYEE ON PROJECT.ID_EMPLOYEE = EMPLOYEE.ID_EMPLOYEE "
        + "ORDER BY PROJECT.ID_PROJECT ASC");
    while (rs.next()) {
        Employee emp = new Employee();
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));

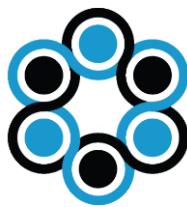
        Job job = new Job();
        job.setJobName(rs.getString("JOB_NAME"));

        Project project = new Project();
        project.setPrjId(rs.getInt("ID_PROJECT"));
        project.setJob(job);
        project.setEmployee(emp);
        project.setStartDate(rs.getDate("START_DATE"));
        project.setDueDate(rs.getDate("END_DATE"));

        this.arrProject.add(project);
    }
    return this.arrProject;
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getProjectData()** yang berfungsi untuk membuat mengambil data-data dari **table Project** yang juga menerapkan query JOIN untuk menampilkan **EMPLOYEE_NAME** dan **JOB_NAME** pada **table Employee** dan **table Job..**



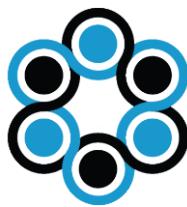
Tugas Resmi

Langkah Ketigabelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **insertEmployee()**, **insertClient()**, **insertJob()**, **insertProject()**.

Query

```
public void insertEmployee(Employee emp) {
    this.conn.ManipulasiData("INSERT INTO
EMPLOYEE VALUES ('" + emp.getEmpId() + "', '" +
emp.getEmpName() + "', '" + emp.getEmpAddr() + "', '" +
+ emp.getEmpGender() + "', '" + emp.getEmpPhoneNo() + "
', '" + emp.getEmpEmail() + "')");
}
public void insertClient(Client cl) {
    this.conn.ManipulasiData("INSERT INTO CLIENT
VALUES ('" + cl.getClId() + "', '" + cl.getClName() +
"', '" + cl.getClAddr() + "', '" + cl.getClEmail() +
"', '" + cl.getClPhoneNo() + "')");
}
public void insertJob(Job job) {
    this.conn.ManipulasiData("INSERT INTO JOB
VALUES ('" + job.getJobId() + "', '" +
job.getClient().getClId() + "', '" + job.getJobName() +
"', '" + job.getJobCat() + "', '" +
job.getJobDesc() + "')");
}
public void insertProject(Project project) {
    try {
        String startDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getStar
tDate());
        String dueDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getDueD
ate());
        this.conn.ManipulasiData("INSERT INTO
PROJECT VALUES (ID_PROJECT.NEXTVAL, " +
project.getJob().getJobId() + ", " +
project.getEmployee().getEmpId() + ", TO_DATE('" +
startDate + "', 'dd/MM/yyyy'), TO_DATE('" + dueDate + "
', 'dd/MM/yyyy'))");
    } catch (Exception e) {
        System.out.println(e);
    }
}
```



Tugas Resmi

Screenshot

```
public void insertEmployee(Employee emp) {
    this.conn.ManipulasiData("INSERT INTO EMPLOYEE VALUES ('" + emp.getEmpId() + "', '" +
        + emp.getEmpName() + "', '" + emp.getEmpAddr() + "', '" +
        + emp.getEmpGender() + "', '" + emp.getEmpPhoneNo() + "', '" +
        + emp.getEmpEmail() + "')");
}

public void insertClient(Client cl) {
    this.conn.ManipulasiData("INSERT INTO CLIENT VALUES ('" + cl.getCliId() + "', '" +
        + cl.getClName() + "', '" + cl.getClAddr() + "', '" +
        + cl.getClEmail() + "', '" +
        + cl.getClPhoneNo() + "')");
}

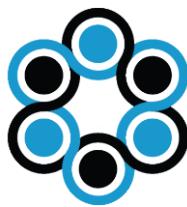
public void insertJob(Job job) {
    this.conn.ManipulasiData("INSERT INTO JOB VALUES ('" + job.getJobId() + "', '" +
        + job.getClient().getCliId() + "', '" + job.getJobName() + "', '" +
        + job.getJobCat() + "', '" +
        + job.getJobDesc() + "')");
}

public void insertProject(Project project) {
    try {
        String startDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getStartDate());
        String dueDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getDueDate());
        this.conn.ManipulasiData("INSERT INTO PROJECT VALUES (ID_PROJECT.NEXTVAL, " +
            + project.getJob().getJobId() + ", " +
            + project.getEmployee().getEmpId() +
            + ", TO_DATE('" + startDate + "', 'dd/MM/yyyy'), TO_DATE('" + dueDate + "', 'dd/MM/yyyy'))");
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

Analisa :

Source code di atas adalah source code **Controller.java** pada bagian method yang digunakan untuk melakukan eksekusi CREATE / INSERT DATA pada table Employee, Client, Job, dan Project. Bisa dilihat pada gambar terdapat beberapa method, yaitu :

- insertEmployee(Employee emp)
- insertClient(Clienet cl)
- insertJob(Job job)
- insertProject(Project project)



Tugas Resmi

Langkah Keempatbelas

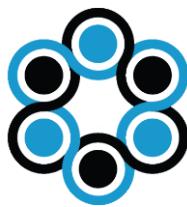
Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **updateEmployee()**, **updateClient()**, **updateJob()**, **updateProject()**.

Query

```
public void updateEmployee(Employee emp) {
    this.conn.ManipulasiData("UPDATE EMPLOYEE SET
EMPLOYEE_NAME = '" + emp.getEmpName()
+ "', EMPLOYEE_ADDR = '" +
emp.getEmpAddr()
+ "', GENDER = '" +
emp.getEmpGender()
+ "', PHONE_NO = '" +
emp.getEmpPhoneNo()
+ "', EMAIL = '" + emp.getEmpEmail()
+ "' WHERE ID_EMPLOYEE = " +
emp.getEmpId());
}

public void updateClient(Client cl) {
    this.conn.ManipulasiData("UPDATE CLIENT SET
ID_CLIENT = " + cl.getClId()
+ ", CLIENT_NAME = '" +
cl.getClName()
+ "', CLIENT_ADDR = '" +
cl.getClAddr()
+ "', CLIENT_EMAIL = '" +
cl.getClEmail()
+ "', CLIENT_PHONE_NO = '" +
cl.getClPhoneNo()
+ "' WHERE ID_CLIENT = " +
cl.getClId());
}

public void updateJob(Job j) {
    this.conn.ManipulasiData("UPDATE JOB SET
ID_CLIENT = " + j.getClient().getClId()
+ ", JOB_NAME = '" + j.getJobName()
+ "', CATEGORY = '" + j.getJobCat()
+ "', DESCRIPTION = '" +
j.getJobDesc()
+ "' WHERE ID_JOB = " +
j.getJobId());
```



Tugas Resmi

```
}
```

```
public void updateProject(Project project) {
    String startDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getStar
tDate());
    String dueDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getDueD
ate());
    this.conn.ManipulasiData("UPDATE PROJECT SET
ID_JOB = " + project.getJob().getJobId()
        + ", ID_EMPLOYEE = " +
project.getEmployee().getEmpId()
        + ", START_DATE = TO_DATE('" +
startDate
        + "', 'dd/MM/yyyy'), END_DATE =
TO_DATE('" + dueDate
        + "', 'dd/MM/yyyy') WHERE ID_PROJECT
= " + project.getPrjId());
}
```

Screenshot

```
public void updateEmployee(Employee emp) {
    this.conn.ManipulasiData("UPDATE EMPLOYEE SET EMPLOYEE_NAME = '" + emp.getEmpName()
        + "', EMPLOYEE_ADDR = '" + emp.getEmpAddr()
        + "', GENDER = '" + emp.getEmpGender()
        + "', PHONE_NO = '" + emp.getEmpPhoneNo()
        + "', EMAIL = '" + emp.getEmpEmail()
        + "' WHERE ID_EMPLOYEE = " + emp.getEmpId());
}

public void updateClient(Client cl) {
    this.conn.ManipulasiData("UPDATE CLIENT SET ID_CLIENT = " + cl.getCliId()
        + ", CLIENT_NAME = '" + cl.getClName()
        + "', CLIENT_ADDR = '" + cl.getClAddr()
        + "', CLIENT_EMAIL = '" + cl.getClEmail()
        + "', CLIENT_PHONE_NO = '" + cl.getClPhoneNo()
        + "' WHERE ID_CLIENT = " + cl.getClId());
}

public void updateJob(Job j) {
    this.conn.ManipulasiData("UPDATE JOB SET ID_CLIENT = " + j.getClient().getClId()
        + ", JOB_NAME = '" + j.getJobName()
        + "', CATEGORY = '" + j.getJobCat()
        + "', DESCRIPTION = '" + j.getJobDesc()
        + "' WHERE ID_JOB = " + j.getJobId());
}
```



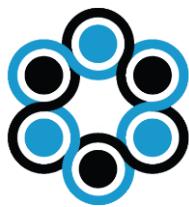
Tugas Resmi

```
public void updateProject(Project project) {  
    String startDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getStartDate());  
    String dueDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getDueDate());  
    this.conn.ManipulasiData("UPDATE PROJECT SET ID_JOB = " + project.getJob().getJobId()  
        + ", ID_EMPLOYEE = " + project.getEmployee().getEmpId()  
        + ", START_DATE = TO_DATE('" + startDate  
        + "', 'dd/MM/yyyy'), END_DATE = TO_DATE('" + dueDate  
        + "', 'dd/MM/yyyy') WHERE ID_PROJECT = " + project.getPrjId());  
}
```

Analisa :

Source code di atas adalah source code **Controller.java** pada bagian method yang digunakan untuk melakukan eksekusi UPDATE DATA pada table Employee, Client, Job, dan Project. Bisa dilihat pada gambar terdapat beberapa method, yaitu:

- updateEmployee(Employee emp)
- updateClient(Clienet cl)
- updateJob(Job j)
- updateProject(Project project)



Tugas Resmi

Langkah Kelimabelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **deleteEmployee()**, **deleteClient()**, **deleteJob()**, **deleteProject()**.

Query

```
public void deleteEmployee(Integer idEmp) {
    this.conn.ManipulasiData("DELETE FROM
EMPLOYEE WHERE ID_EMPLOYEE = " + idEmp);
}
public void deleteClient(Integer idClient) {
    this.conn.ManipulasiData("DELETE FROM CLIENT
WHERE ID_CLIENT = " + idClient);
}
public void deleteJob(Integer idJob) {
    this.conn.ManipulasiData("DELETE FROM JOB
WHERE ID_JOB = " + idJob);
}
public void deleteProject(Integer Id) {
    this.conn.ManipulasiData("DELETE FROM PROJECT
WHERE ID_PROJECT = " + Id);
}
```

Screenshot

```
public void deleteEmployee(Integer idEmp) {
    this.conn.ManipulasiData("DELETE FROM EMPLOYEE WHERE ID_EMPLOYEE = " + idEmp);
}

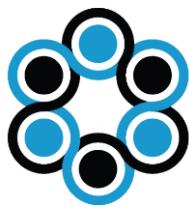
public void deleteClient(Integer idClient) {
    this.conn.ManipulasiData("DELETE FROM CLIENT WHERE ID_CLIENT = " + idClient);
}

public void deleteJob(Integer idJob) {
    this.conn.ManipulasiData("DELETE FROM JOB WHERE ID_JOB = " + idJob);
}

public void deleteProject(Integer Id) {
    this.conn.ManipulasiData("DELETE FROM PROJECT WHERE ID_PROJECT = " + Id);
}
```

Analisa :

Source code di atas adalah source code **Controller.java** pada bagian method yang digunakan untuk melakukan eksekusi **DELETE DATA** pada table Employee, Client, Job, dan Project.



Tugas Resmi

Langkah Keenambelas

Pindah ke class **OutsourcingCompany.java** , class tersebut adalah main class pada aplikasi ini.

Query

```
package com.outsourcingcompany.main;

import com.outsourcingcompany.view.*;
import java.sql.SQLException;
import java.text.ParseException;

public class OutsourcingCompany {
    public static void main(String[] args) throws
SQLException, ParseException {
        // TODO code application logic here
        new LoginView().show();
    }
}
```

Screenshot

```
package com.outsourcingcompany.main;

import com.outsourcingcompany.view.*;
import java.sql.SQLException;
import java.text.ParseException;

public class OutsourcingCompany {
    public static void main(String[] args) throws SQLException, ParseException {
        // TODO code application logic here
        new LoginView().show();
    }
}
```

Analisa :

Source code di atas adalah source code dari **OutsourcingCompany.java** yang merupakan main class dari aplikasi ini. Class ini berfungsi untuk memanggil class **LoginView.java** pada package View yang berisi semua class JFrame / GUI.



Tugas Resmi

Langkah Ketujuhbelas

Membuat class JFrame dengan nama **LoginView.java**.

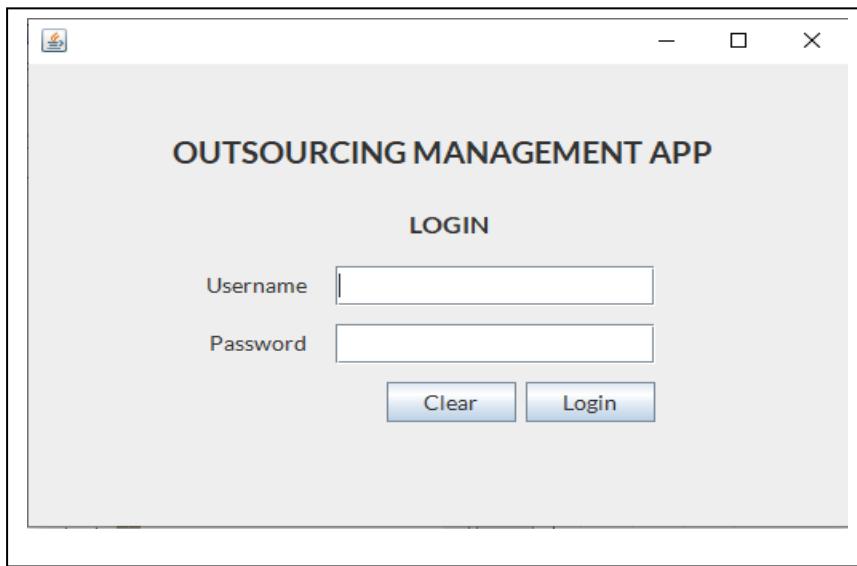
Query

```
private void  
btnLoginOkActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String username =  
txtUsername.getText().trim();  
    String password =  
txtPassword.getText().trim();  
  
    if(username.equals("admin") &&  
password.equals("admin123")) {  
        try {  
            new HomeAdminView().show();  
            setVisible(false);  
        } catch (SQLException ex) {  
  
Logger.getLogger(LoginView.class.getName()).log(Level.SEVERE, null, ex);  
        }  
    } else if(username.equals("hrd") &&  
password.equals("hrd123")) {  
        try {  
            new ProjectView().show();  
            setVisible(false);  
        } catch (SQLException ex) {  
  
Logger.getLogger(LoginView.class.getName()).log(Level.SEVERE, null, ex);  
        }  
    } else {  
  
JOptionPane.showMessageDialog(null, "WRONG  
PASSWORD!");  
        setVisible(false);  
    }  
}
```



Tugas Resmi

Screenshot



Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **LoginView.java** yang berfungsi untuk membuat halaman login. Username yang disediakan adalah sebagai berikut:

Login Admin

Username : **admin**

Password : **admin123**

Login HRD

Username : **hrd**

Password : **hrd123**



Tugas Resmi

Langkah Kedelapanbelas

Membuat class JFrame dengan nama **HomeAdminView.java**.

Query

```
private void
btnEntryPegawaiActionPerformed(java.awt.event.
ActionEvent evt) {
    // TODO add your handling code here:
    try {
        new EntryEmployeeView().show();
    } catch (SQLException e) {

Logger.getLogger(LoginView.class.getName()).lo
g(Level.SEVERE, null, e);
    }
}

private void
btnEntryClientActionPerformed(java.awt.event.A
ctionEvent evt) {
    // TODO add your handling code here:
    try {
        new EntryClientView().show();
    } catch (SQLException e) {

Logger.getLogger(LoginView.class.getName()).lo
g(Level.SEVERE, null, e);
    }
}

private void
btnEntryJobActionPerformed(java.awt.event.Acti
onEvent evt) {
    // TODO add your handling code here:
    try {
        new EntryJobView().show();
    } catch (SQLException e) {

Logger.getLogger(LoginView.class.getName()).lo
g(Level.SEVERE, null, e);
    }
}
```



Tugas Resmi

```
private void
btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int ans =
JOptionPane.showOptionDialog(this,
        "Do you want to logout?",
        "Logout",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE,
null, null, null);

    if (ans == JOptionPane.YES_OPTION) {

JOptionPane.showMessageDialog(null, "Logout
succeeded!");
try {
    new LoginView().show();
    setVisible(false);

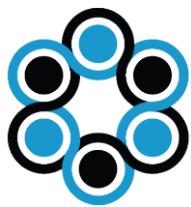
} catch (SQLException ex) {

Logger.getLogger(HomeAdminView.class.getName()
).log(Level.SEVERE, null, ex);
}

    }
}
```

Screenshot

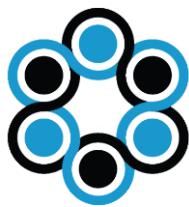




Tugas Resmi

Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **HomeAdminView.java** yang berfungsi sebagai menu bagi Admin apakah ingin melakukan transaksi data terhadap Employee, Client, atau Job. Terdapat button untuk logout juga jika admin sudah atau tidak akan melakukan transaksi data terhadap tiga kategori tersebut.



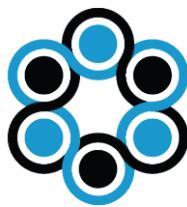
Tugas Resmi

Langkah Kesembilanbelas

Membuat class JFrame dengan nama **EntryEmployeeView.java**.

Query

```
private void  
btnAddEmpActionPerformed(java.awt.event.ActionEvent  
evt) {  
    // TODO add your handling code here:  
    try {  
        Employee emp = new Employee();  
  
        emp.setEmpId(Integer.parseInt(this.txtEmpId.ge  
tText()));  
  
        emp.setEmpName(this.txtEmpName.getText());  
  
        emp.setEmpAddr(this.txtEmpAddr.getText());  
  
        emp.setEmpGender(this.cbEmpGender.getSelecte  
dItem().toString());  
  
        emp.setEmpPhoneNo(this.txtEmpPhone.getText());  
  
        emp.setEmpEmail(this.txtEmpEmail.getText());  
  
        this.controller.insertEmployee(emp);  
  
        JOptionPane.showMessageDialog(null, "Employee  
added!");  
        this.showTableEmployeeData();  
    } catch (SQLException ex) {  
  
        Logger.getLogger(EntryEmployeeView.class.getNa  
me()).log(Level.SEVERE, null, ex);  
    }  
  
}  
  
private void  
btnUpdateEmpActionPerformed(java.awt.event.Act
```



Tugas Resmi

```
private void
btnUpdateEmpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbEmployeeData.getValueAt(this.tbEmployee
Data.getSelectedRow(), 0).toString();
        Employee emp = new Employee();

        emp.setEmpId(Integer.parseInt(selected));

        emp.setEmpName(this.txtEmpName.getText());

        emp.setEmpAddr(this.txtEmpAddr.getText());

        emp.setEmpGender(this.cbEmpGender.getSelectedItem()
.toString());

        emp.setEmpPhoneNo(this.txtEmpPhone.getText());

        emp.setEmpEmail(this.txtEmpEmail.getText());

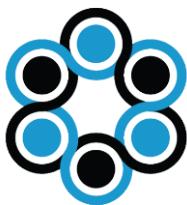
        this.controller.updateEmployee(emp);

        JOptionPane.showMessageDialog(null, "Employee
updated!");
        this.showTableEmployeeData();

    } catch (SQLException e) {

        Logger.getLogger(EntryEmployeeView.class.getName())
.log(Level.SEVERE, null, e);
    }
}

private void
btnDelEmpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbEmployeeData.getValueAt(this.tbEmployee
Data.getSelectedRow(), 0).toString();
    }
}
```



Tugas Resmi

```
Employee emp = new Employee();
emp.setEmpId(Integer.parseInt(selected));

emp.setEmpName(this.txtEmpName.getText());
emp.setEmpAddr(this.txtEmpAddr.getText());
emp.setEmpGender(this.cbEmpGender.getSelectedItem().toString());
emp.setEmpPhoneNo(this.txtEmpPhone.getText());
emp.setEmpEmail(this.txtEmpEmail.getText());
this.controller.updateEmployee(emp);

JOptionPane.showMessageDialog(null, "Employee updated!");
this.showTableEmployeeData();

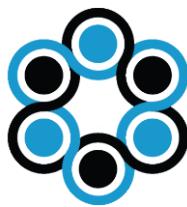
} catch (SQLException e) {
Logger.getLogger(EntryEmployeeView.class.getName()).log(Level.SEVERE, null, e);
}

private void
btnDelEmpActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
try {
String selected =
this.tbEmployeeData.getValueAt(this.tbEmployeeData.getSelectedRow(), 0).toString();

this.controller.deleteEmployee(Integer.parseInt(selected));

JOptionPane.showMessageDialog(null, "Employee deleted!");
this.showTableEmployeeData();

} catch (SQLException ex) {
```



Tugas Resmi

```
Logger.getLogger(EntryEmployeeView.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void
tbEmployeeDataMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    this.txtEmpId.setEditable(false);

    this.txtEmpId.setText(this.tbEmployeeData.getValueAt(this.tbEmployeeData.getSelectedRow(),
0).toString());
}
```

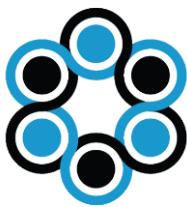
Screenshot

The screenshot shows a Java Swing application window titled "OUTSOURCING MANAGEMENT APP". The window has a title bar with standard minimize, maximize, and close buttons. Inside, there is a panel for "Employee Data Entry" containing input fields for Employee ID, Employee Name, Employee Address, Gender (a dropdown menu), Phone No., and Email, along with three buttons: "Add", "Delete", and "Update". To the right of this panel is a table titled "EMPLOYEE..". The table has columns for EMPLOYEE.., NAME, ADDRESS, GENDER, PHONE NO, and EMAIL. It contains 14 rows of data, each representing an employee record. The data is as follows:

EMPLOYEE..	NAME	ADDRESS	GENDER	PHONE NO	EMAIL
106	Maya	Medaeng	P	1066666	maya@gma...
107	Aga	Waru	L	1077777	aga@gmai...
108	Nara	Sidoarjo	P	1088888	nara@gma...
109	Moka	Candi	L	1099999	moka@gm...
110	Tara	Tanggulan...	L	1100000	tara@gmai...
111	Roni	Sedati	L	1111111	roni@gmai...
112	Yuna	Waru	P	1222222	yuna@gma...
113	Rainy	Sidoarjo	P	1333333	rainy@gma...
114	Moza	Candi	L	1444444	moza@gm...
115	Taro	Tanggulan...	L	1555555	taro@gmai...
101	Muchias	Dusun Dur...	L	8124186	muchias@...
102	Sterling	Surabaya S...	L	8225045	sterling@g...
103	Narendra	Taman Bu...	L	81110145	narendra@...
104	Nurani	Rungkut	P	9993331	nurani@ns...

Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **EntryEmployeeView.java** yang berfungsi sebagai menu bagi Admin untuk melakukan transaksi data (create, update, dan delete) pada **table Employee** yang ada pada database oracle.



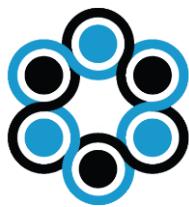
Tugas Resmi

Langkah Keduapuluhan

Membuat class JFrame dengan nama **EntryClientView.java**.

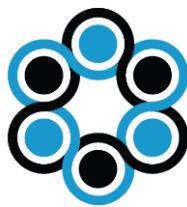
Query

```
private void  
btnAddClientActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try {  
        Client cl = new Client();  
  
        cl.setClId(Integer.parseInt(this.txtClientId.getText()));  
  
        cl.setClName(this.txtClientName.getText());  
  
        cl.setClAddr(this.txtClientAddr.getText());  
  
        cl.setClEmail(this.txtClientEmail.getText());  
  
        cl.setClPhoneNo(this.txtClientPhone.getText())  
;  
        this.controller.insertClient(cl);  
        this.showTableClientData();  
  
    } catch (SQLException ex) {  
  
        Logger.getLogger(EntryClientView.class.getName()  
()).log(Level.SEVERE, null, ex);  
    }  
}  
  
private void  
btnUpdateClActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try {  
        String selected =  
this.tbClientData.getValueAt(this.tbClientData  
.getSelectedRow(), 0).toString();  
        Client cl = new Client();  
  
        cl.setClId(Integer.parseInt(selected));  
    }
```



Tugas Resmi

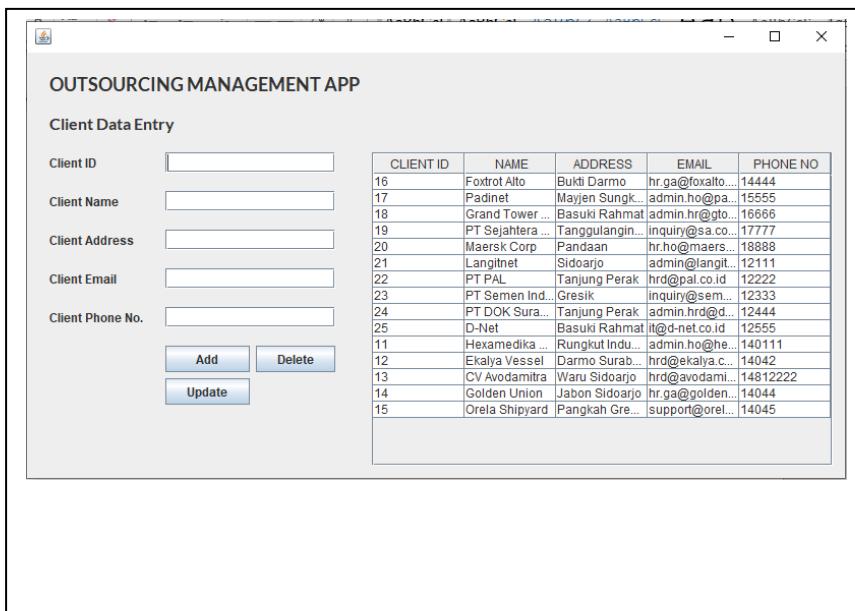
```
cl.setClName(this.txtClientName.getText());  
  
cl.setClAddr(this.txtClientAddr.getText());  
  
cl.setClEmail(this.txtClientEmail.getText());  
  
cl.setClPhoneNo(this.txtClientPhone.getText())  
;  
        this.controller.updateClient(cl);  
        this.showTableClientData();  
  
    } catch (SQLException e) {  
  
        Logger.getLogger(EntryClientView.class.getName()  
()).log(Level.SEVERE, null, e);  
    }  
}  
  
private void  
btnDelClActionPerformed(java.awt.event.ActionEvent  
evt) {  
    // TODO add your handling code here:  
    try {  
        String selected =  
this.tbClientData.getValueAt(this.tbClientData  
.getSelectedRow(), 0).toString();  
  
this.controller.deleteClient(Integer.parseInt(  
selected));  
        this.showTableClientData();  
  
    } catch (SQLException e) {  
  
        Logger.getLogger(EntryClientView.class.getName()  
()).log(Level.SEVERE, null, e);  
    }  
}  
  
private void  
tbClientDataMouseClicked(java.awt.event.MouseE  
vent evt) {  
    // TODO add your handling code here:  
    this.txtClientId.setEditable(false);  
  
this.txtClientId.setText(this.tbClientData.get  
ValueAt(this.tbClientData.getSelectedRow(),
```



Tugas Resmi

```
private void  
tbClientDataMouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    this.txtClientId.setEditable(false);  
  
    this.txtClientId.setText(this.tbClientData.getValueAt(this.tbClientData.getSelectedRow(),  
0).toString());  
}
```

Screenshot



Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **EntryClientView.java** yang berfungsi sebagai menu bagi Admin untuk melakukan transaksi data (create, update, dan delete) pada **table Client** yang ada pada database oracle.



Tugas Resmi

Langkah Keduapuluhsatu

Membuat class JFrame dengan nama **EntryJobView.java**.

Query

```
private void
btnAddJobActionPerformed(java.awt.event.ActionEvent
Event evt) {
    // TODO add your handling code here:
    try {
        Job job = new Job();

        job.setJobId(Integer.parseInt(this.txtJobId.ge
tText()));

        job.setClient(this.controller.getClientData().get(this.cbClientId.getSelectedIndex()));

        job.setJobName(this.txtJobName.getText());

        job.setJobCat(this.cbJobCat.getSelectedItem().to
String());

        job.setJobDesc(this.txtJobDesc.getText());
        this.controller.insertJob(job);

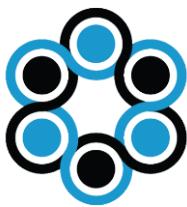
        JOptionPane.showMessageDialog(null, "Job
added!");

        this.showTableJobData();

    } catch (SQLException e) {

Logger.getLogger(EntryJobView.class.getName())
.log(Level.SEVERE, null, e);
    }
}

private void
btnUpdJobActionPerformed(java.awt.event.ActionEvent
Event evt) {
    // TODO add your handling code here:
```



Tugas Resmi

```
private void
btnUpdJobActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    String selected =
this.tbJobData.getValueAt(this.tbJobData.getSelectedRow(), 0).toString();
    Job j = new Job();

    try {

j.setJobId(Integer.parseInt(selected));

j.setClient(this.controller.getClientData().get(this.cbClientId.getSelectedIndex()));

j.setJobName(this.txtJobName.getText());

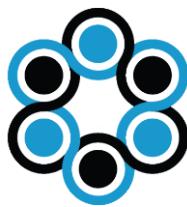
j.setJobCat(this.cbJobCat.getSelectedItem().toString());

j.setJobDesc(this.txtJobDesc.getText());
        this.controller.updateJob(j);

JOptionPane.showMessageDialog(null, "Job
Updated!");
        this.showTableJobData();
    } catch (SQLException ex) {

Logger.getLogger(EntryJobView.class.getName())
.log(Level.SEVERE, null, ex);
    }

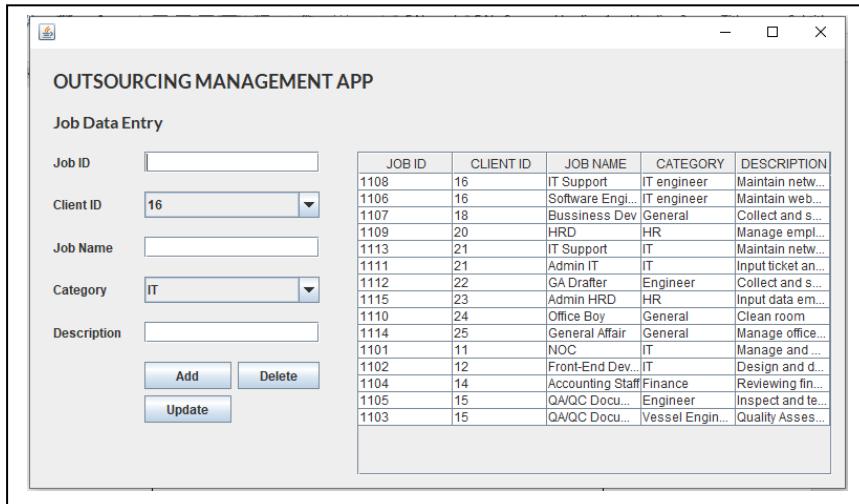
    private void
btnDelJobActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbJobData.getValueAt(this.tbJobData.getSelectedRow(), 0).toString();
```



Tugas Resmi

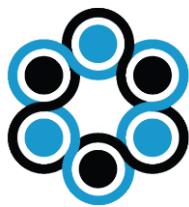
```
this.controller.deleteJob(Integer.parseInt(selected));
        JOptionPane.showMessageDialog(null,
"Job Deleted!");
        this.showTableJobData();
    } catch (SQLException ex) {
Logger.getLogger(EntryJobView.class.getName())
.log(Level.SEVERE, null, ex);
    }
}
private void
tbJobDataMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    this.txtJobId.setEditable(false);
this.txtJobId.setText(this.tbJobData.getValueAt(
this.tbJobData.getSelectedRow(),
0).toString());
    }
```

Screenshot



Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **EntryJobView.java** yang berfungsi sebagai menu bagi Admin untuk melakukan transaksi data (create, update, dan delete) pada **table Job** yang ada pada database oracle.



Tugas Resmi

Langkah Keduapuluuhdua

Membuat class JFrame dengan nama **ProjectView.java**.

Query

```
private void
btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Project project = new Project();
    try {

        project.setJob(this.assignment.getJobData().ge
t(this.cbJobId.getSelectedIndex()));

        project.setEmployee(this.assignment.getEmploye
eData().get(this.cbEmpId.getSelectedIndex()));
        project.setStartDate(new
SimpleDateFormat("dd/MM/yyyy").parse(this.txtS
tartDate.getText()));
        project.setDueDate(new
SimpleDateFormat("dd/MM/yyyy").parse(this.txtD
ueDate.getText()));

        this.assignment.insertProject(project);

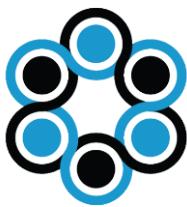
        JOptionPane.showMessageDialog(null, "Project
added!");
        this.showTableProject();

    } catch (SQLException | ParseException
err) {
        System.out.println(err);
    }

}

private void
tblEmployeeListMouseClicked(java.awt.event.Mou
seEvent evt) {
    // TODO add your handling code here:

}
```



Tugas Resmi

```
private void
tbOutstandingJobMouseClicked(java.awt.event.Mo
useEvent evt) {
    // TODO add your handling code here:
}

private void
btnUpdateActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    try {
        String selectedRow =
this.tbOnProgressProject.getValueAt(this.tbOnP
rogressProject.getSelectedRow(),
0).toString();
        Project project = new Project();

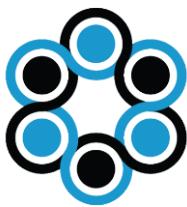
        project.setPrjId(Integer.parseInt(selectedRow)
);

        project.setJob(this.assignment.getJobData().ge
t(this.cbJobId.getSelectedIndex()));

        project.setEmployee(this.assignment.getEmploye
eData().get(this.cbEmpId.getSelectedIndex()));
        project.setStartDate(new
SimpleDateFormat("dd/MM/yyyy").parse(this.txtS
tartDate.getText()));
        project.setDueDate(new
SimpleDateFormat("dd/MM/yyyy").parse(this.txtD
ueDate.getText()));

        this.assignment.updateProject(project);

        JOptionPane.showMessageDialog(null, "Project
updated!");
        this.showTableProject();
    } catch (SQLException | ParseException
err) {
        System.out.println(err);
    }
}
```



Tugas Resmi

```
private void
btnDelActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbOnProgressProject.getValueAt(this.tbOnP
rogressProject.getSelectedRow(),
0).toString();

        this.assignment.deleteProject(Integer.parseInt
(selected));
JOptionPane.showMessageDialog(null, "Project
deleted!");
        this.showTableProject();
    } catch (NumberFormatException | 
SQLException e) {

Logger.getLogger(ProjectView.class.getName()).
log(Level.SEVERE, null, e);
    }
}

private void
btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int ans =
JOptionPane.showOptionDialog(this,
        "Do you want to logout?",
        "Logout",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE,
null, null, null);

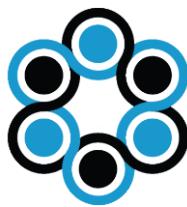
    if (ans == JOptionPane.YES_OPTION) {

JOptionPane.showMessageDialog(null, "You have
been logged out!");
        try {
            new LoginView().show();
            setVisible(false);

        } catch (SQLException ex) {

Logger.getLogger(HomeAdminView.class.getName())

```

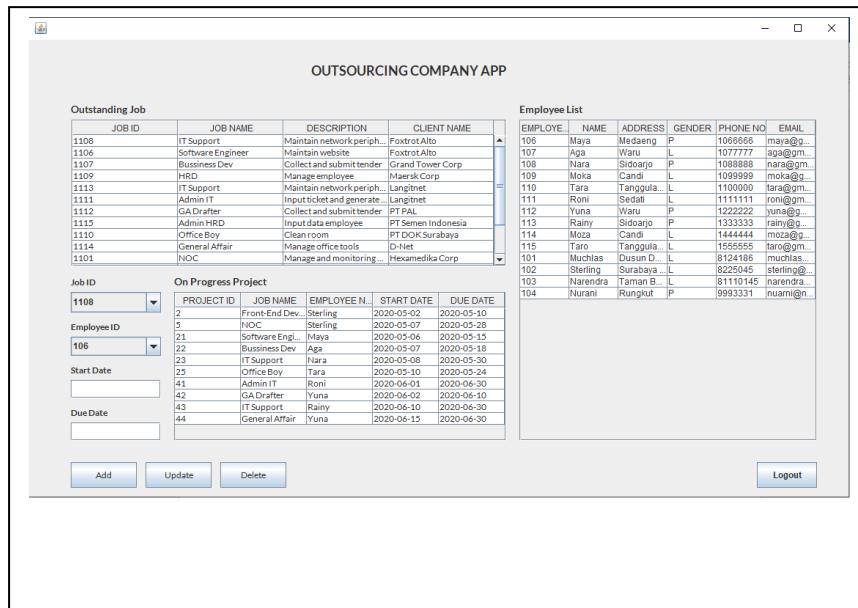


Tugas Resmi

```
    ).log(Level.SEVERE, null, ex);
}

}
```

Screenshot



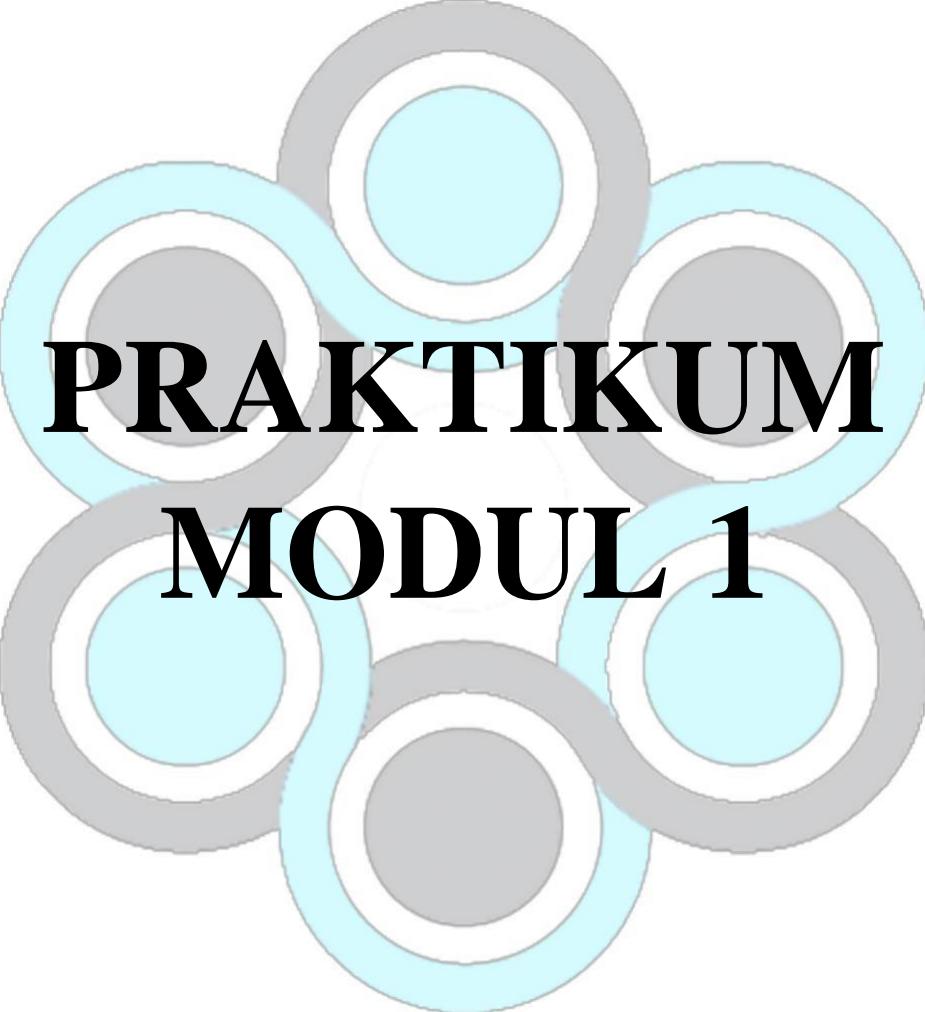
Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **ProjectView.java** yang berfungsi sebagai menu bagi HRD untuk melakukan transaksi data (create, update, dan delete) pada **table Project** yang ada pada database oracle.



LAPORAN PRAKTIKUM

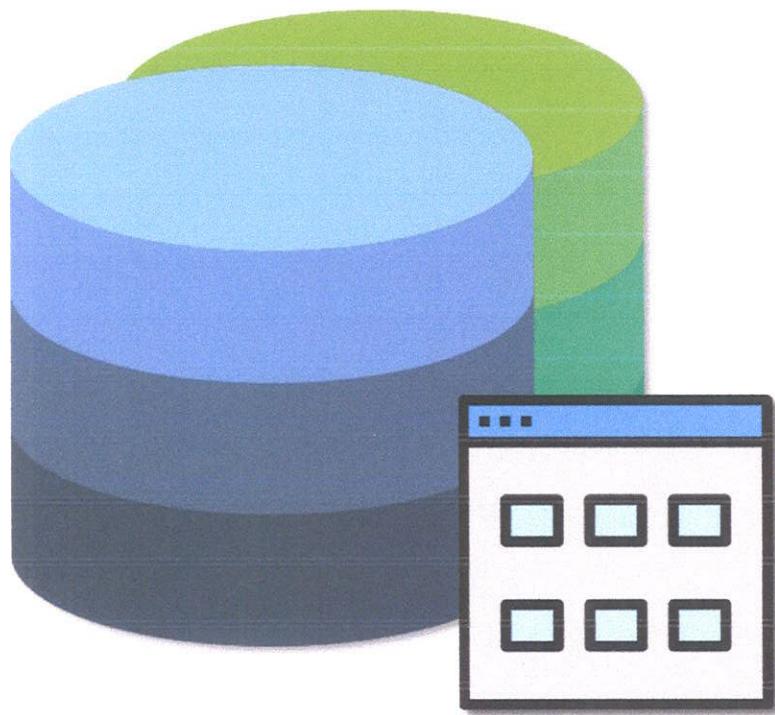
FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



PRAKTIKUM MODUL 1

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020

LAPORAN PRAKTIKUM

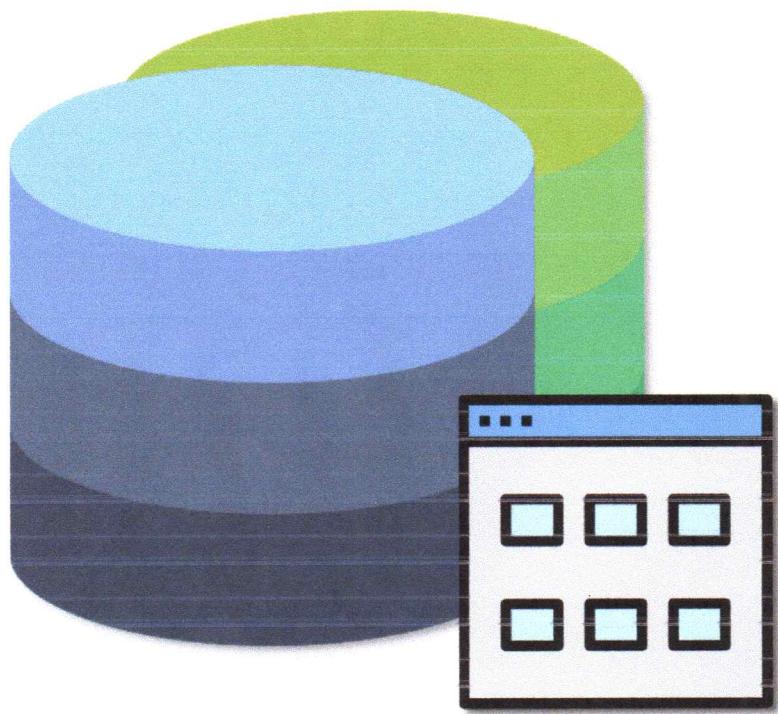


PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmad Muchlasin
NPM : 06.2018 - 1 - 06941
MODUL : 1

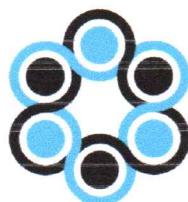
--

LAPORAN PENDAHULUAN



PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmad Muchlasin
NPM : 06.2018.1.06941
MODUL : I



Tugas Pertanyaan

Modul 1

Bab I , Bab II

1.1 Tujuan

1. Membuat struktur database berdasarkan PAM

1.2 Kriteria Penilaian

1. Pembuatan table dan perubahan struktur table (DDL) berdasarkan PAM.

1.3 Soal Pendahuluan

Perhatikan Diagram berikut

1. Buatlah Sistem Informasi Penjualan sederhana dimana terdapat 8 tabel yang harus saling berelasi, yaitu tabel Pegawai, Pelanggan, Kurs, Pemasok, Kategori, Produk, Pesanan dan Retail Pesanan.

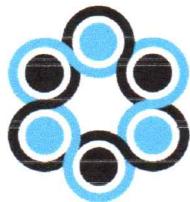
Lanjutkan Pertama

- Buatlah tablespace seperti contoh dibawah ini :

```
CREATE TABLESPACE <nama-tablespace>
DATAFILE 'nama_file-tablespace.dbf'
SIZE "ukuran-table space";
```

```
sql> CREATE TABLESPACE sistem_jual
2  datafile 'D:\ITATS\Matkul\semester 4\Basi\data\'
3  Praktikum\sistem-jual.dbf'
3.  size 30M;
```

Tablespace created .



Tugas Pertanyaan

- Buat user dengan kotentuan ITAMA-NINDA

CREATE USER < nama-user >

IDENTIFIED BY < password-user >

DEFAULT TABLESPACE < nama-tablespace >

QUOTA <batas_ukuran-tablespace> ON <nama-tablespace>;

SQL > CREATE USER jannie_012345

2 IDENTIFIED BY jannie

3 DEFAULT TABLESPACE sistemjul

4 QUOTA 30M ON sistemjul;

User created.

- Beri hak akses "All Privileges" untuk user

GRANT ALL PRIVILEGES TO jannie_012345;

SQL > GRANT ALL PRIVILEGES TO jannie_012345;

Grant succeeded.

- Login ke user yang telah dibuat

CONN < nama-user > /< password-user >

SQL > conn jannie_012345 / jannie

Connected.

Langkah Kedua

Buat tabel pegawai dengan nama <pegawai-NPM> dan

berberapa kolom id-pegawai, nama-depan, nama-belakang,

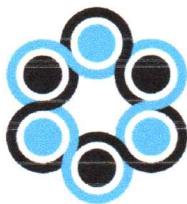
tanggal_lahir, alamat, kota_kerja dan no_hp. Dimana id_pegawai

sebagai Primary Key .

SQL > create table Pegawai_012345

2

3 id_pegawai INTEGER not null,



Tugas Pertanyaan

```
4. nama-depan      varchar(15),  
5. nama-belakang  varchar(15),  
6. tanggal-lahir    DATE,  
7. alamat          varchar(100),  
8. kode-pes         number(7),  
9. no-telp          number(10),  
10. constraint pk-pegawai primary key (id-pegawai)  
11. );
```

Table created.

Laylah Ketiga

Buat tabel Pelanggan dengan name < pelanggan-NPM > dan beberapa kolom id-pelanggan, nama-depan, nama-belakang, tanggal-lelah, dant, kode-pes, dan no-telp. Dimana id-pelanggan, sebagai Primary Key.

SQL > create table pelanggan - 02345

2. (

3. id-pelanggan INTEGER not null,

4. nama-depan varchar2(15),

5. nama-belakang varchar2(15),

6. tanggal-lelah DATE,

7. alamat varchar2(100),

8. kode-pes number(7),

9. no-tlp number(10),

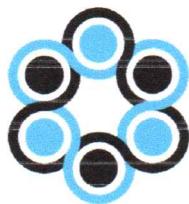
10. constraint pk-pelanggan primary key (id-pelanggan)

11.);

Table created.

Laylah Keempat

Buat tabel Kurir dengan name < kurir-NPM > dan



Tugas Pertanyaan

beberapa kolom id_kurir, nama_perusahaan dan no_telp.

Dinamai id_kurir, sebagai Primary Key.

SQL > create tabel kurir - crs45

2 (

3 id_kurir INTEGER notnull,

4 nama_perusahaan varchar(16),

5 no_telp number(12),

6. constraint PK_kurir primary key (id_kurir).

7)i

Table created.

Langkah Kelima

Buat tabel pemesanan dengan nama < pemesan_NPM > dan
kolom id_pemesan, nama_perusahaan, alamat, kode_pos
dan no_telp. Dinamai id_pemesan sebagai Primary Key.

SQL > create table pemesan - crs45

2 (

3 id_pemesan INTEGER notnull,

4 nama_perusahaan varchar(15),

5 alamat varchar(20),

6 kode_pos number(7),

7 no_telp number(12),

8 constraint PK_pemesan primary key (id_pemesan)

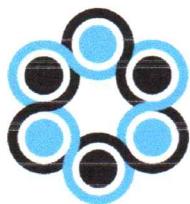
9)i

Table created.

Langkah Keenam

Buat tabel Kategori dengan nama < kategori_NPM > dan
kolom id_kategori, nama_kategori, dan deskripsi.

Dinamai id_kategori sebagai Primary Key.



Tugas Pertanyaan

SQL > create table kategori_arsys

```
2 (
3   id_kategori  INTEGER  not null,
4   nama_kategori  varchar2(16),
5   deskripsi  varchar2(20),
6   constraint PK_kategori primary key (id_kategori)
7 );
```

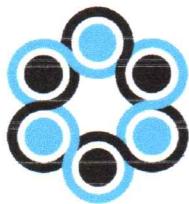
Table created.

Langkah Ketiga

Buat tabel Produk dengan nama < produk_NPM > dan beberapa kolom id_produk, id_pemasok, id_kategori, nama_produk, harga_satuan dan stok_produk. Dimana id_produk sebagai primary key, id_pemasok, dan id_kategori sebagai foreign key.

SQL > create table produk_arsys

```
2 (
3   id_produk  INTEGER  not null,
4   id_pemasok  INTEGER,
5   id_kategori  INTEGER,
6   nama_produk  varchar2(50),
7   harga_satuan  number(9),
8   stok_produk  number(3),
9   constraint PK_produk primary key (id_produk),
10  constraint FK_pemasok foreign key (id_pemasok),
11    REFERENCES Pemasok_arsys (id_pemasok),
12  constraint FK_Kategori foreign key (id_kategori),
13    REFERENCES kategori_arsys (id_kategori)
14 ); Table created.
```



Tugas Pertanyaan

Langkah Kedelapan

Buat tabel pemesanan dengan nama `transaksi - MPT1` dan berisi kolom `id-pemesanan`, `id-pelanggan`, `id-pekerjaan`, `id-kurir`, `terganteng-pemesanan`, `terganteng-pengiriman`, `alamat-pengiriman`, dan `harga-totol`.

Dimana `id-pemesanan` dengan nama `transaksi` primary key.

SQL > Create table `Pemesanan - 012345`

2 C

3 `id-pemesanan` INTEGER NOT NULL,

4 `id-pelanggan` INTEGER,

5 `id-pekerjaan` INTEGER,

6 `id-kurir` INTEGER,

7 `terganteng-pemesanan` DATE,

8 `terganteng-pengiriman` DATE,

9 `alamat-pengiriman` varchar(100),

10 `harga-totol` number(9),

11 constraint PK-pemesanan primary key
(`id-pemesanan`)

Table created.

Langkah Kesembilan

Tambahkan Foreign Key ke tabel pemesanan pada kolom `id-pelanggan`, `id-pekerjaan`, `id-kurir`, dengan menggunakan Alter Statement.

SQL > alter table `Pemesanan - 012345`

2 add constraint FK_id_Pelanggan FOREIGN KEY (`id-pelanggan`)

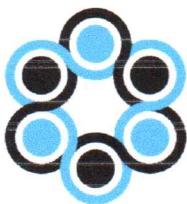
3 references Pelanggan - 012345 (`id-pelanggan`)

4 add constraint FK_id_pekerjaan FOREIGN KEY (`id-pekerjaan`)

5 references Pekerjaan - 012345 (`id-pekerjaan`)

6 add constraint FK_id_kurir FOREIGN KEY (`id-kurir`)

7 references Kurir - 012345 (`id-kurir`); Table altered.



Tugas Pertanyaan

Langkah Kesebelas

Buat tabel Detail_Pemesanan dengan beberapa kolom
id-produk, id-pemesanan, jumlah, dan diskon.

SQL > create table detail_pemesanan_012345

```
2 (
3     id-produk    INTEGER    not null,
4     id-pemesanan    INTEGER,
5     jumlah    number(3),
6     diskon    float(4)
7 );
```

Table created.

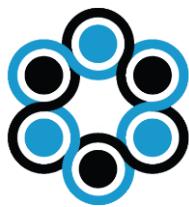
Langkah kesebelus

Tambahkan Foreign Key ke tabel Detail_Pemesanan
pada kolom id-produk dan id-pemesanan dengan
menggunakan Alter statement.

SQL > alter table detail-pemesanan_012345

```
2     add constraint FK_Id-produk FOREIGN KEY (id-produk)
3     references Produk_012345 (id-produk)
4     add constraint FK_Id-pemesanan FOREIGN KEY
5     references Pemesanan_012345 (id-pemesanan);
```

Table altered.



Tugas Praktikum

Soal Praktikum

1. Buatlah user dan tablespace sesuai Nama dan NPM masing – masing praktikan.
(done)
2. Buatlah tabel sesuai Desain Database Proyek Kelas yang sudah di ACC oleh dosen. (done)
3. Buatlah relasi antar tabel sesuai Desain Database dengan perintah ALTER.
(done)
4. Menerapkan Sequence pada salah satu tabel. (done)
5. Ubah nama salah satu field pada salah satu tabel dengan menambah NPM didepan nama field.
Contoh : 06852_NamaProduk
6. Ubah salah satu field pada salah satu tabel dengan Attribute Constraint UNIQUE.
7. Ubah salah satu field pada salah satu tabel dengan Tipe data yang berbeda.

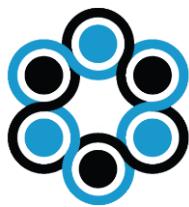
Langkah Pertama

Membuat tablespace dan user sesuai Nama dan NPM menggunakan salah satu perintah DDL yaitu **CREATE TABLESPACE** dan **CREATE USER**. Login dengan menggunakan user **system**.

Query

```
CREATE TABLESPACE muchlasin_06941
datafile 'C:\Users\Achmad
Muchlasin\praktikum_basdat\outsourcing_comp.d
bf'
size 30M;

CREATE USER muchlasin_06941
IDENTIFIED BY muchlas
DEFAULT TABLESPACE muchlasin_06941
QUOTA 30M ON muchlasin_06941;
```



Tugas Praktikum

Screenshot

```
SQL> CREATE TABLESPACE muchlasin_06941
  2  datafile 'C:\Users\Achmad Muchlasin\praktikum_basdat\outsourcing_comp.dbf'
  3  size 30M;

Tablespace created.
```

```
SQL> CREATE USER muchlasin_06941
  2  IDENTIFIED BY muchlas
  3  DEFAULT TABLESPACE muchlasin_06941
  4  QUOTA 30M ON muchlasin_06941;

User created.
```

Analisa

CREATE TABLESPACE

Pada perintah CREATE TABLESPACE terdapat syntax tambahan yaitu:

1) Datafile

Digunakan untuk menentukan lokasi tablespace yang dibuat.

2) Size

Digunakan untuk menentukan ukuran dari tablespace yang telah dibuat.

CREATE USER

Pada perintah CREATE TABLESPACE terdapat syntax tambahan yaitu:

1) Identified by

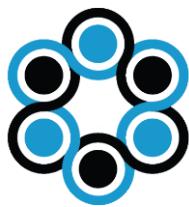
Digunakan untuk meng-set password dari user yang telah dibuat

2) Default tablespace

Digunakan untuk meng-set default tablespace untuk yang yang telah dibuat

3) Quota

Digunakan untuk menentukan kuota yang bisa digunakan oleh user dalam sebuah tablespace.



Tugas Praktikum

Langkah Kedua

Setelah membuat TABLESPACE dan USER maka langkah selanjutnya adalah menentukan privileges dari user yang telah kita buat dengan menggunakan perintah **GRANT**. Lalu login dengan user yang telah kita buat tadi dengan perintah **CONN**.

Query

```
GRANT ALL PRIVILEGES TO muchlasin_06941;  
CONN muchlasin_06941/muchlas
```

Screenshot

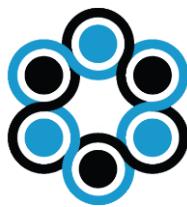
```
SQL> GRANT ALL PRIVILEGES TO muchlasin_06941;  
Grant succeeded.  
  
SQL> CONN muchlasin_06941/muchlas  
Connected.
```

Analisa :

Karena pada langkah paling awal kita masih menggunakan user **system**. Jadi, agar kita bisa membuat privileges yang sama seperti **system** untuk user baru kita maka kita perlu menggunakan perintah **GRANT ALL PRIVILEGES**. Saat login dengan menggunakan perintah **CONN** maka user baru kita hak aksesnya sudah sama seperti **system** tadi.

Langkah Ketiga

Membuat table sesuai dengan desain yang telah kita buat. Yang pertama kita akan membuat table **Employee**. Dengan beberapa kolom yaitu, **id_employee**, **employee_name**, **employee_addr**, dll.



Tugas Praktikum

Query

```
CREATE TABLE employee(
    id_employee int not null,
    employee_name VARCHAR2(50),
    employee_addr VARCHAR2(100),
    gender VARCHAR2(5),
    phone_no number(12),
    email number(30),
    CONSTRAINT PK_employee PRIMARY KEY
(id_employee)
);  
  
DESC employee
```

Screenshot

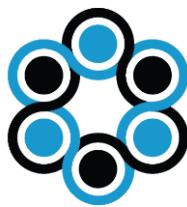
```
SQL> CREATE TABLE employee(
 2      id_employee int not null,
 3      employee_name VARCHAR2(50),
 4      employee_addr VARCHAR2(100),
 5      gender VARCHAR2(5),
 6      phone_no number(12),
 7      email number(30),
 8      CONSTRAINT PK_employee PRIMARY KEY (id_employee)
 9  );  
Table created.
```

```
SQL> desc employee;
Name          Null?    Type
ID_EMPLOYEE      NOT NULL NUMBER(38)
EMPLOYEE_NAME           VARCHAR2(50)
EMPLOYEE_ADDR          VARCHAR2(100)
GENDER                VARCHAR2(5)
PHONE_NO              NUMBER(12)
EMAIL                 NUMBER(30)  
SQL>
```

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Employee berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks



Tugas Praktikum

CONSTRAINT untuk menentukan field mana yang merupakan **PRIMARY KEY** (id_employee). Dan field yang dibuat di dalam table Employee adalah :

- **id_employee (PK)** INTEGER
- employee_name VARCHAR2(50)
- employee_addr VARCHAR2(100)
- gender VARCHAR2(5)
- phone_no NUMBER(12)
- email NUMBER(30)

Pada Gambar #2 :

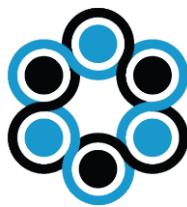
Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_employee**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan PRIMARY KEY.

Langkah Keempat

Langkah selanjutnya adalah kita akan membuat table **Client**. Dengan beberapa kolom yaitu, id_client, client_name, client_addr, dll.

Query

```
CREATE TABLE client(  
    id_client int not null,  
    client_name VARCHAR2(50),  
    client_addr VARCHAR2(100),  
    client_email varchar(30),  
    client_phone_no number(12),  
    CONSTRAINT PK_client PRIMARY KEY  
    (id_client)  
);  
  
DESC client
```



Tugas Praktikum

Screenshot

```
SQL> CREATE TABLE client(
  2      id_client int not null,
  3      client_name VARCHAR2(50),
  4      client_addr VARCHAR2(100),
  5      client_email varchar(30),
  6      client_phone_no number(12),
  7      CONSTRAINT PK_client PRIMARY KEY (id_client)
  8  );  
Table created.
```

```
SQL> DESC client  
Name          Null?    Type  
-----  
ID_CLIENT      NOT NULL NUMBER(38)  
CLIENT_NAME    VARCHAR2(50)  
CLIENT_ADDR    VARCHAR2(100)  
CLIENT_EMAIL   VARCHAR2(30)  
CLIENT_PHONE_NO NUMBER(12)  
SQL>
```

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Client berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks **CONSTRAINT** untuk menentukan field mana yang merupakan **PRIMARY KEY** (`id_client`). Dan field yang dibuat di dalam table Client adalah :

- **id_client (PK)** INTEGER
- **client_name** VARCHAR2(50)
- **client_addr** VARCHAR2(100)
- **client_email** VARCHAR2(30)
- **client_phone_no** NUMBER(12)

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_client**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan **PRIMARY KEY**.



Tugas Praktikum

Langkah Kelima

Langkah selanjutnya adalah kita akan membuat table **Job**. Dengan beberapa kolom yaitu, id_job, id_client, job_name, dll.

Query

```
CREATE TABLE job(
    id_job int not null,
    id_client int,
    job_name VARCHAR2(100),
    category VARCHAR2(50),
    description VARCHAR2(100),
    CONSTRAINT PK_job PRIMARY KEY (id_job)
);
DESC job
```

Screenshot

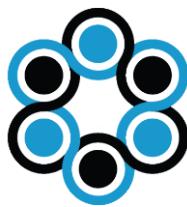
```
SQL> CREATE TABLE job(
  2      id_job int not null,
  3      id_client int,
  4      job_name VARCHAR2(100),
  5      category VARCHAR2(50),
  6      description VARCHAR2(100),
  7      CONSTRAINT PK_job PRIMARY KEY (id_job)
  8  );
```

```
Table created.
```

```
SQL>
```

```
SQL> DESC job
Name          Null?    Type
-----          -----
ID_JOB        NOT NULL NUMBER(38)
ID_CLIENT     NUMBER(38)
JOB_NAME      VARCHAR2(100)
CATEGORY      VARCHAR2(50)
DESCRIPTION   VARCHAR2(100)
```

```
SQL>
```



Tugas Praktikum

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Job berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks **CONSTRAINT** untuk menentukan field mana yang merupakan **PRIMARY KEY** (**id_job**). Dan field yang dibuat di dalam table Employee adalah :

- id_job (PK)	INTEGER
- id_client (FK)	INTEGER
- job_name	VARCHAR2(100)
- category	VARCHAR2(50)
- description	VARCHAR2(100)

Pada Gambar #2 :

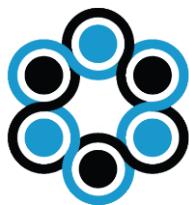
Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_job**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan **PRIMARY KEY**.

Langkah Keenam

Langkah selanjutnya adalah kita akan membuat table **Project**. Dengan beberapa kolom yaitu, **id_project**, **id_job**, **id_employee**, dll.

Query

```
CREATE TABLE project(
    id_project int not null,
    id_job int,
    id_employee int,
    start_date DATE,
    end_date DATE,
    CONSTRAINT PK_project PRIMARY KEY
```



Tugas Praktikum

```
 );
DESC project
```

Screenshot

```
SQL> CREATE TABLE project(
  2      id_project int not null,
  3      id_job int,
  4      id_employee int,
  5      start_date DATE,
  6      end_date DATE,
  7      CONSTRAINT PK_project PRIMARY KEY (id_project)
  8  );
```

Table created.

```
SQL> DESC project
Name          Null?    Type
-----        -----   -----
ID_PROJECT    NOT NULL NUMBER(38)
ID_JOB         NUMBER(38)
ID_EMPLOYEE   NUMBER(38)
START_DATE    DATE
END_DATE      DATE
```

SQL>

Analisa :

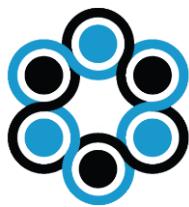
Pada Gambar #1 :

Dapat dilihat query untuk membuat table Client berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks **CONSTRAINT** untuk menentukan field mana yang merupakan **PRIMARY KEY** (`id_project`). Dan field yang dibuat di dalam table Employee adalah :

- **id_employee (PK)** INTEGER
- **id_job (FK)** INTEGER
- **id_employee (FK)** INTEGER
- **start_date** DATE
- **end_date** DATE

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL pada field **id_employee**, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan **PRIMARY KEY**.



Tugas Praktikum

Langkah Ketujuh

Langkah selanjutnya adalah kita akan membuat relasi antar table dengan menggunakan perintah ALTER.

Query

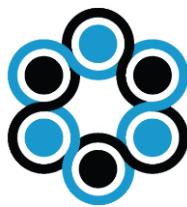
```
ALTER TABLE job
add CONSTRAINT FK_job_client FOREIGN KEY
(id_client)
REFERENCES client (id_client);

ALTER TABLE project
add CONSTRAINT FK_project_employee FOREIGN
KEY (id_employee)
REFERENCES employee(id_employee)
add CONSTRAINT FK_project_job FOREIGN KEY
(id_job)
REFERENCES job(id_job);
```

Screenshot

```
SQL> ALTER TABLE job
2 add CONSTRAINT FK_job_client FOREIGN KEY (id_client)
3 REFERENCES client (id_client);
Table altered.

SQL> ALTER TABLE project
2 add CONSTRAINT FK_project_employee FOREIGN KEY (id_employee)
3 REFERENCES employee(id_employee)
4 add CONSTRAINT FK_project_job FOREIGN KEY (id_job)
5 REFERENCES job(id_job);
Table altered.
```



Tugas Praktikum

Analisa :

Pada Gambar #1 :

Dapat dilihat relasi yang dibuat antara table job dengan client sudah berhasil dengan menggunakan perintah **add CONSTRAINT**. Terdapat juga nama alias **FK_job_client** yang merupakan nama dari **CONSTRAINT FOREIGN KEY** yang kita buat.

Pada Gambar #2 :

Dapat dilihat relasi yang dibuat antara table employee dengan table project dan juga table job dengan table project sudah berhasil dengan menggunakan perintah **add CONSTRAINT**. Fungsi nama alias untuk **CONSTRAINT** yang dibuat sangatlah berguna untuk membedakan **FOREIGN KEY** dari masing-masing table pada database Oracle.

Langkah Kedelapan

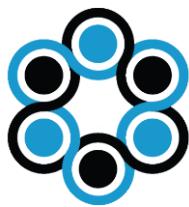
Langkah selanjutnya adalah kita membuat sequence yaitu dengan menggunakan perintah **CREATE SEQUENCE**.

Query

```
CREATE SEQUENCE id_project  
MINVALUE 1  
MAXVALUE 100  
START WITH 1  
INCREMENT BY 1  
CACHE 20;
```

Screenshot

```
SQL> CREATE SEQUENCE id_project  
2 MINVALUE 1  
3 MAXVALUE 100  
4 START WITH 1  
5 INCREMENT BY 1  
6 CACHE 20;  
  
Sequence created.
```



Tugas Praktikum

Analisa :

Pada gambar di atas, saya membuat sequence untuk table Project dimana field yang menggunakan sequence tersebut adalah **id_project**. Keterangan dari sintaks yang lain juga bisa disimak di bawah ini:

- minvalue : sintaks untuk menentukan nilai terkecil pada sequence.
- maxvalue : sintaks untuk menentukan nilai terakhir pada sequence.
- start with : sintaks untuk menentukan nilai awal pada sequence.
- increment by : sintaks untuk menentukan nilai di tiap pertambahannya.

Langkah Kesembilan

Langkah selanjutnya adalah kita merubah nama field sesuai ketentuan. Perintah yang digunakan yaitu **ALTER – RENAME COLUMN**.

Query

```
ALTER TABLE employee RENAME COLUMN  
id_employee to 06941_id_employee;
```

Screenshot

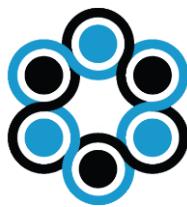
```
SQL> ALTER TABLE employee RENAME COLUMN employee_name TO 06941_employee_name;  
Table altered.
```

Analisa :

Pada gambar di atas nama field sudah berhasil saya ubah sesuai permintaan.

Langkah Kesepuluh

Langkah selanjutnya adalah kita merubah tipe constraint. Pada langkah ini saya menggunakan field yang ada pada table Employee yaitu **phone_no**. Perintah yang digunakan adalah **ALTER - ADD**.



Tugas Praktikum

Query

```
ALTER TABLE employee  
add UNIQUE (phone_no);
```

Screenshot

```
SQL> ALTER TABLE employee  
2 add UNIQUE (phone_no);  
Table altered.  
SQL> DESC employee  
Name Null? Type  
----  
ID_EMPLOYEE NOT NULL NUMBER(38)  
06941_EMPLOYEE_NAME VARCHAR2(50)  
EMPLOYEE_ADDR VARCHAR2(100)  
GENDER VARCHAR2(5)  
PHONE_NO NUMBER(12)  
EMAIL NUMBER(30)  
SQL>
```

Analisa :

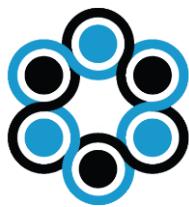
Pada gambar di atas menunjukkan bahwa salah satu field yang ada pada table Employee yaitu **phone_no** saya rubah tipe constraint nya dengan menggunakan perintah **ALTER** ditambah dengan **add UNIQUE**. Saya menggunakan field **phone_no** karena tiap orang pasti memiliki no telfon yang berbeda-beda. Perbedaan antara **UNIQUE** dan **PRIMARY KEY** adalah **PRIMARY KEY** selalu bersifat unique namun **UNIQUE** belum tentu **PRIMARY KEY**. Karna constraint **UNIQUE** biasanya digunakan untuk field yang memiliki keunikan di tiap datanya.

Langkah Kesembilan

Langkah selanjutnya adalah kita merubah tipe data dari field dengan menggunakan perintah **ALTER – MODIFY**.

Query

```
ALTER TABLE employee MODIFY(email  
VARCHAR2(30));
```



Tugas Praktikum

Screenshot

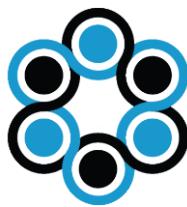
```
SQL> ALTER TABLE employee MODIFY(email VARCHAR2(30));
Table altered.

SQL> DESC employee
Name          Null?    Type
-----        -----
ID_EMPLOYEE   NOT NULL NUMBER(38)
06941_EMPLOYEE_NAME  VARCHAR2(50)
EMPLOYEE_ADDR  VARCHAR2(100)
GENDER         VARCHAR2(5)
PHONE_NO       NUMBER(12)
EMAIL          VARCHAR2(30)

SQL>
```

Analisa :

Pada gambar di atas nama salah satu field tipe nya sudah saya ubah menjadi VARCHAR



Laporan Sementara

Soal Praktikum

1. Buatlah user dan tablespace sesuai Nama dan NPM masing – masing praktikan.
(done)
2. Buatlah tabel sesuai Desain Database Proyek Kelas yang sudah di ACC oleh dosen. (done)
3. Buatlah relasi antar tabel sesuai Desain Database dengan perintah ALTER.
(done)
4. Menerapkan Sequence pada salah satu tabel. (done)
5. Ubah nama salah satu field pada salah satu tabel dengan menambah NPM didepan nama field.
Contoh : 06852_NamaProduk
6. Ubah salah satu field pada salah satu tabel dengan Attribute Constraint UNIQUE.
7. Ubah salah satu field pada salah satu tabel dengan Tipe data yang berbeda.

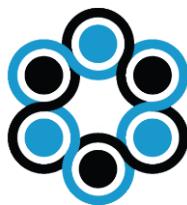
Langkah Pertama

Membuat tablespace dan user sesuai Nama dan NPM menggunakan salah satu perintah DDL yaitu **CREATE TABLESPACE** dan **CREATE USER**. Login dengan menggunakan user **system**.

Query

```
CREATE TABLESPACE muchlasin_06941
datafile 'C:\Users\Achmad
Muchlasin\praktikum_basdat\outsourcing_comp.db
f'
size 30M;

CREATE USER muchlasin_06941
IDENTIFIED BY muchlas
DEFAULT TABLESPACE muchlasin_06941
QUOTA 30M ON muchlasin_06941;
```



Laporan Sementara

Screenshot

```
SQL> CREATE TABLESPACE muchlasin_06941
  2  datafile 'C:\Users\Achmad Muchlasin\praktikum_basdat\outsourcing_comp.dbf'
  3  size 30M;

Tablespace created.
```

```
SQL> CREATE USER muchlasin_06941
  2  IDENTIFIED BY muchlas
  3  DEFAULT TABLESPACE muchlasin_06941
  4  QUOTA 30M ON muchlasin_06941;

User created.
```

Analisa

CREATE TABLESPACE

Pada perintah CREATE TABLESPACE terdapat syntax tambahan yaitu:

1) Datafile

Digunakan untuk menentukan lokasi tablespace yang dibuat.

2) Size

Digunakan untuk menentukan ukuran dari tablespace yang telah dibuat.

CREATE USER

Pada perintah CREATE TABLESPACE terdapat syntax tambahan yaitu:

1) Identified by

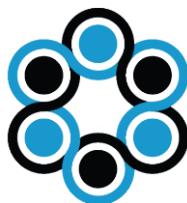
Digunakan untuk meng-set password dari user yang telah dibuat

2) Default tablespace

Digunakan untuk meng-set default tablespace untuk yang yang telah dibuat

3) Quota

Digunakan untuk menentukan kuota yang bisa digunakan oleh user dalam sebuah tablespace.



Laporan Sementara

Langkah Kedua

Setelah membuat TABLESPACE dan USER maka langkah selanjutnya adalah menentukan privileges dari user yang telah kita buat dengan menggunakan perintah **GRANT**. Lalu login dengan user yang telah kita buat tadi.

Query

```
GRANT ALL PRIVILEGES TO muchlasin_06941;  
CONN muchlasin_06941/muchlas
```

Screenshot

```
SQL> GRANT ALL PRIVILEGES TO muchlasin_06941;  
Grant succeeded.  
  
SQL> CONN muchlasin_06941/muchlas  
Connected.
```

Analisa :

Karena pada langkah paling awal kita masih menggunakan user **system**. Jadi, agar kita bisa membuat privileges yang sama seperti system untuk user baru kita maka kita perlu menggunakan perintah **GRANT ALL PRIVILEGES**. Saat login dengan menggunakan perintah **CONN** maka user baru kita hak aksesnya sudah sama seperti **system** tadi.



Laporan Sementara

Langkah Ketiga

Membuat table sesuai dengan desain yang telah kita buat. Yang pertama kita akan membuat table **Employee**. Dengan beberapa kolom yaitu, id_employee, employee_name, employee_addr, dll.

Query

```
CREATE TABLE employee(
    id_employee int not null,
    employee_name VARCHAR2(50),
    employee_addr VARCHAR2(100),
    gender VARCHAR2(5),
    phone_no number(12),
    email number(30),
    CONSTRAINT PK_employee PRIMARY KEY
(id_employee)
);
```



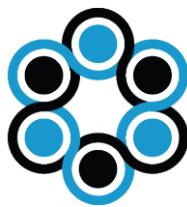
```
DESC employee
```

Screenshot

```
SQL> CREATE TABLE employee(
 2   id_employee int not null,
 3   employee_name VARCHAR2(50),
 4   employee_addr VARCHAR2(100),
 5   gender VARCHAR2(5),
 6   phone_no number(12),
 7   email number(30),
 8   CONSTRAINT PK_employee PRIMARY KEY (id_employee)
 9 );
```

Table created.

```
SQL> desc employee;
Name          Null?    Type
-----          -----
ID_EMPLOYEE          NOT NULL NUMBER(38)
EMPLOYEE_NAME        VARCHAR2(50)
EMPLOYEE_ADDR        VARCHAR2(100)
GENDER              VARCHAR2(5)
PHONE_NO            NUMBER(12)
EMAIL               NUMBER(30)
```



Laporan Sementara

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Employee berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks untuk menentukan field mana yang merupakan PRIMARY KEY (id_employee).

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan PRIMARY KEY.

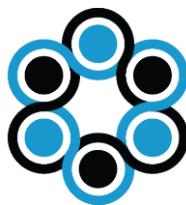
Langkah Keempat

Langkah selanjutnya adalah kita akan membuat table **Client**. Dengan beberapa kolom yaitu, id_client, client_name, client_addr, dll.

Query

```
CREATE TABLE client(
    id_client int not null,
    client_name VARCHAR2(50),
    client_addr VARCHAR2(100),
    client_email varchar(30),
    client_phone_no number(12),
    CONSTRAINT PK_client PRIMARY KEY
    (id_client)
);

DESC client
```



Laporan Sementara

Screenshot

```
SQL> CREATE TABLE client(
  2      id_client int not null,
  3      client_name VARCHAR2(50),
  4      client_addr VARCHAR2(100),
  5      client_email varchar(30),
  6      client_phone_no number(12),
  7      CONSTRAINT PK_client PRIMARY KEY (id_client)
  8  );
```

Table created.

```
SQL> DESC client
Name          Null?    Type
-----        -----
ID_CLIENT      NOT NULL NUMBER(38)
CLIENT_NAME    VARCHAR2(50)
CLIENT_ADDR    VARCHAR2(100)
CLIENT_EMAIL   VARCHAR2(30)
CLIENT_PHONE_NO NUMBER(12)
```

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Client berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks untuk menentukan field mana yang merupakan PRIMARY KEY (id_client).

Pada Gambar #2 :

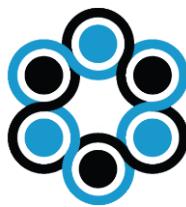
Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan PRIMARY KEY.

Langkah Kelima

Langkah selanjutnya adalah kita akan membuat table **Job**. Dengan beberapa kolom yaitu, id_job, id_client, job_name, dll.

Query

```
CREATE TABLE job(
  id_job int not null,
  id_client int,
```



Laporan Sementara

```
job_name VARCHAR2(100),  
category VARCHAR2(50),  
description VARCHAR2(100),  
CONSTRAINT PK_job PRIMARY KEY (id_job)  
);  
  
DESC job
```

Screenshot

```
SQL> CREATE TABLE job(  
 2      id_job int not null,  
 3      id_client int,  
 4      job_name VARCHAR2(100),  
 5      category VARCHAR2(50),  
 6      description VARCHAR2(100),  
 7      CONSTRAINT PK_job PRIMARY KEY (id_job)  
 8  );
```

```
Table created.
```

```
SQL>
```

```
SQL> DESC job  
Name          Null?    Type  
-----  
ID_JOB        NOT NULL NUMBER(38)  
ID_CLIENT     NUMBER(38)  
JOB_NAME      VARCHAR2(100)  
CATEGORY      VARCHAR2(50)  
DESCRIPTION   VARCHAR2(100)  
SQL>
```

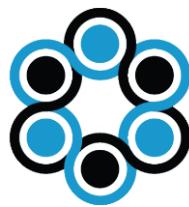
Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Job berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks untuk menentukan field mana yang merupakan PRIMARY KEY (id_job).

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan PRIMARY KEY.



Laporan Sementara

Langkah Keenam

Langkah selanjutnya adalah kita akan membuat table **Project**. Dengan beberapa kolom yaitu, id_project, id_job, id_employee, dll.

Query

```
CREATE TABLE project(
    id_project int not null,
    id_job int,
    id_employee int,
    start_date DATE,
    end_date DATE,
    CONSTRAINT PK_project PRIMARY KEY
(id_project)
);
```

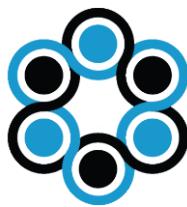
Screenshot

```
SQL> CREATE TABLE project(
  2      id_project int not null,
  3      id_job int,
  4      id_employee int,
  5      start_date DATE,
  6      end_date DATE,
  7      CONSTRAINT PK_project PRIMARY KEY (id_project)
  8  );
```

Table created.

```
SQL> DESC project
Name          Null?    Type
ID_PROJECT    NOT NULL NUMBER(38)
ID_JOB        NUMBER(38)
ID_EMPLOYEE   NUMBER(38)
START_DATE    DATE
END_DATE     DATE
```

SQL>



Laporan Sementara

Analisa :

Pada Gambar #1 :

Dapat dilihat query untuk membuat table Client berhasil dibuat. Tidak ada error yang muncul saat membuat table. Dan pada akhir query ditambahkan sintaks untuk menentukan field mana yang merupakan PRIMARY KEY (id_project).

Pada Gambar #2 :

Terlihat dengan jelas table beserta field yang kita buat sudah sesuai dengan query yang kita gunakan. Dan terdapat keterangan NOT NULL, yang berarti field tersebut tidak boleh kosong karena field tersebut merupakan PRIMARY KEY.

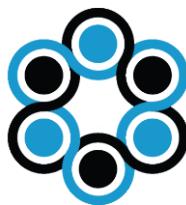
Langkah Ketujuh

Langkah selanjutnya adalah kita akan membuat relasi antar table dengan menggunakan perintah ALTER.

Query

```
ALTER TABLE job
add CONSTRAINT FK_job_client FOREIGN KEY
(id_client)
REFERENCES client (id_client);

ALTER TABLE project
add CONSTRAINT FK_project_employee FOREIGN
KEY (id_employee)
REFERENCES employee(id_employee)
add CONSTRAINT FK_project_job FOREIGN KEY
(id_job)
REFERENCES job(id_job);
```



Laporan Sementara

Screenshot

```
SQL> ALTER TABLE job
  2  add CONSTRAINT FK_job_client FOREIGN KEY (id_client)
  3  REFERENCES client (id_client);

Table altered.
```

```
SQL> ALTER TABLE project
  2  add CONSTRAINT FK_project_employee FOREIGN KEY (id_employee)
  3  REFERENCES employee(id_employee)
  4  add CONSTRAINT FK_project_job FOREIGN KEY (id_job)
  5  REFERENCES job(id_job);

Table altered.
```

Analisa :

Pada Gambar #1 :

Dapat dilihat relasi yang dibuat antara table job dengan client sudah berhasil.

Pada Gambar #2 :

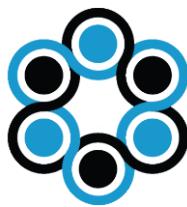
Dapat dilihat relasi yang dibuat antara table employee dengan project dan job dengan project sudah berhasil.

Langkah Kedelapan

Langkah selanjutnya adalah kita membuat sequence

Query

```
CREATE SEQUENCE id_project
MINVALUE 1
MAXVALUE 100
START WITH 1
INCREMENT BY 1
CACHE 20;
```



Laporan Sementara

Screenshot

```
SQL> CREATE SEQUENCE id_project
  2  MINVALUE 1
  3  MAXVALUE 100
  4  START WITH 1
  5  INCREMENT BY 1
  6  CACHE 20;
Sequence created.
```

Analisa :

Pada gambar sequence saya set untuk table project yang digunakan oleh field id_project.

Langkah Kesembilan

Langkah selanjutnya adalah kita merubah nama field sesuai ketentuan.

Query

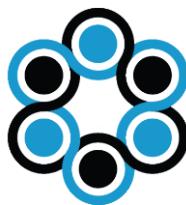
```
ALTER TABLE employee RENAME COLUMN
id_employee to 06941_id_employee;
```

Screenshot

```
SQL> ALTER TABLE employee RENAME COLUMN employee_name TO 06941_employee_name;
Table altered.
```

Analisa :

Pada gambar di atas nama field sudah berhasil saya ubah sesuai permintaan.



Laporan Sementara

Langkah Kesembilan

Langkah selanjutnya adalah kita merubah nama field sesuai ketentuan.

Query

```
ALTER TABLE employee  
add UNIQUE (phone_no);
```

Screenshot

```
SQL> ALTER TABLE employee  
  2 add UNIQUE (phone_no);  
Table altered.  
SQL> DESC employee  
Name          Null?    Type  
-----  
ID_EMPLOYEE      NOT NULL NUMBER(38)  
06941_EMPLOYEE_NAME      VARCHAR2(50)  
EMPLOYEE_ADDR      VARCHAR2(100)  
GENDER      VARCHAR2(5)  
PHONE_NO      NUMBER(12)  
EMAIL      NUMBER(30)  
SQL>
```

Analisa :

Pada gambar di atas nama salah satu field constraintnya sudah saya ubah menjadi UNIQUE

Langkah Kesembilan

Langkah selanjutnya adalah kita merubah nama field sesuai ketentuan.

Query

```
ALTER TABLE employee MODIFY(email  
VARCHAR2(30));
```



Laporan Sementara

Screenshot

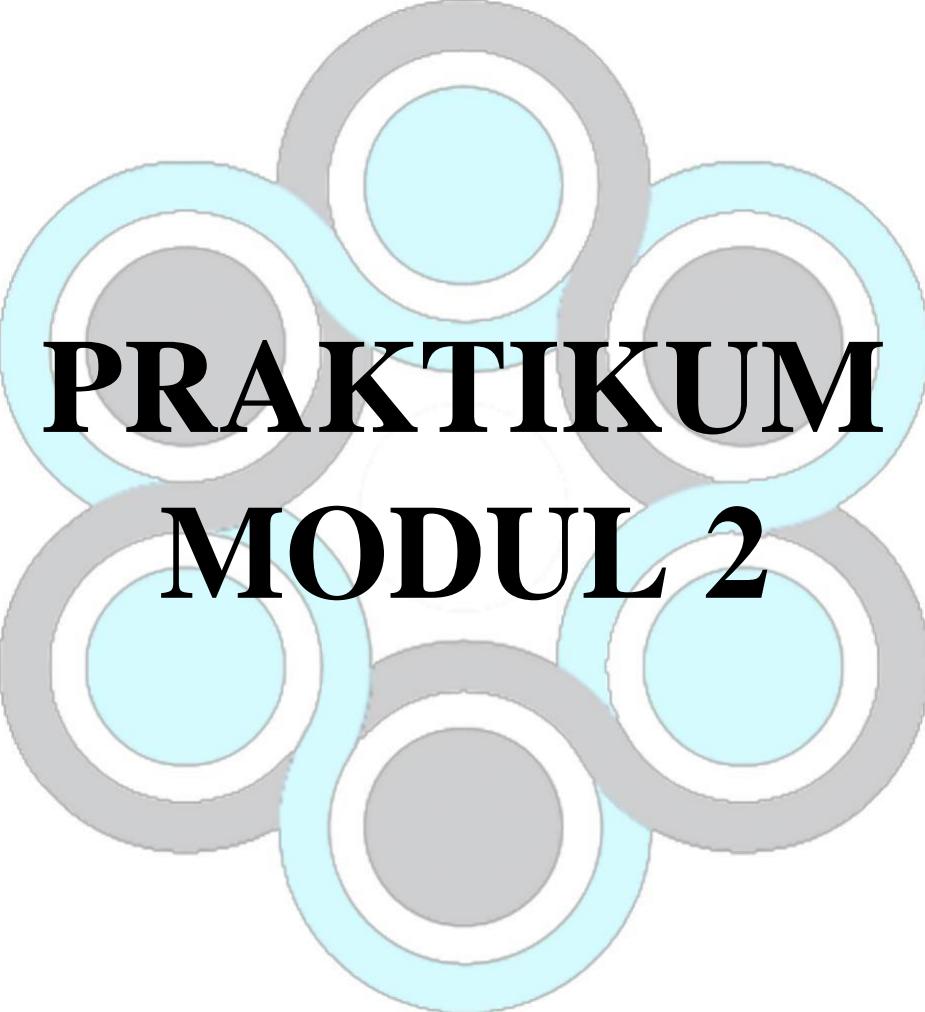
```
SQL> ALTER TABLE employee MODIFY(email VARCHAR2(30));
Table altered.

SQL> DESC employee
Name          Null?    Type
-----        -----
ID_EMPLOYEE      NOT NULL NUMBER(38)
06941_EMPLOYEE_NAME  VARCHAR2(50)
EMPLOYEE_ADDR    VARCHAR2(100)
GENDER           VARCHAR2(5)
PHONE_NO         NUMBER(12)
EMAIL            VARCHAR2(30)

SQL>
```

Analisa :

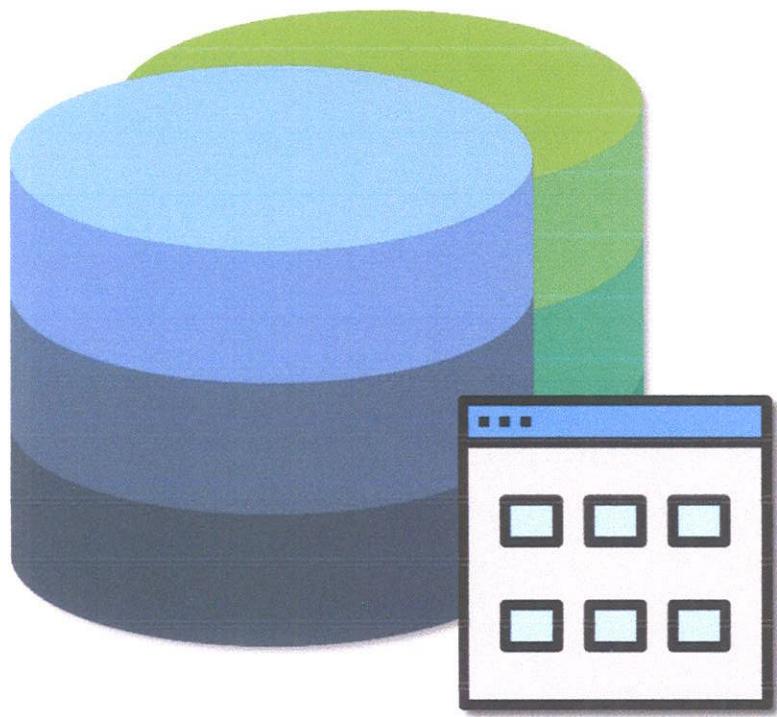
Pada gambar di atas nama salah satu field tipe nya sudah saya ubah menjadi VARCHAR



PRAKTIKUM MODUL 2

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020

LAPORAN PRAKTIKUM

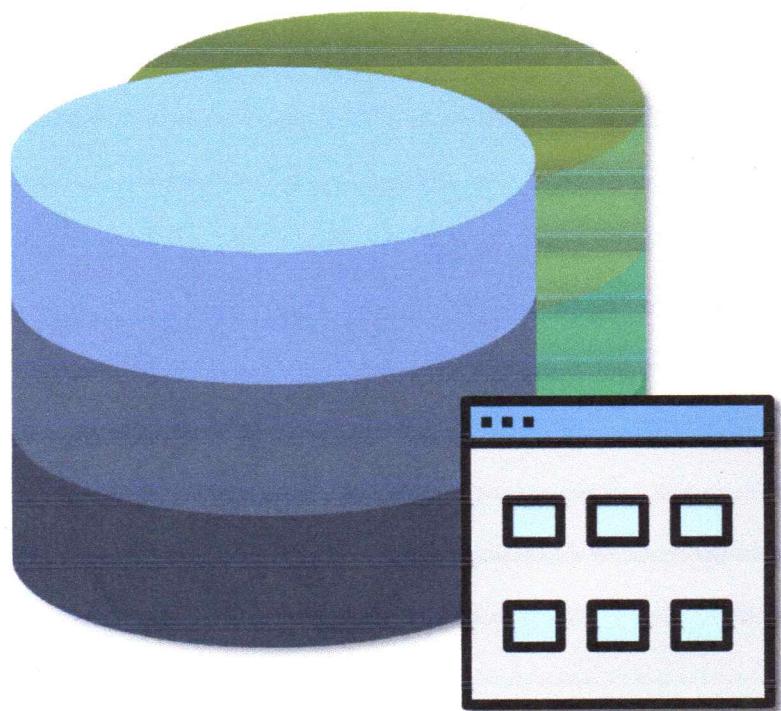


PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmad Muchlasin
NPM : 06.2018-1-06941
MODUL : II

--

LAPORAN PENDAHULUAN



PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmad Muchlasin
NPM : 06.2018.1.06941
MODUL : II



Tugas Pertanyaan

Modul II

BAB III

2.1 Tujuan

- Praktikan dapat mengimplementasikan konsep HTML pada database.

2.2 Kriteria Penilaian

- Praktikan dapat melakukan insert, update dan delete data.
- Praktikan dapat menampilkan data dengan menggunakan klause, create, alias, group by, order by, dan having.

2.3 Soal Pendahuluan

- Insert semua table berdasarkan PBMT pada Modul I dengan syarat: 8 baris data pada table Pegawai, Pelanggan, Kurir, Remasok, Kategori, Produk, Pemesanan dan detail kemasanan.

- Tabel Pegawai

SQL > insert all

```
2 into pegawai_012345 (id_pegawai, nama_depan,  
nama_belakang, tanggal_lahir, alamat, kode_pos,  
no_telp) values (1, 'Bejo', 'Sibjo', to_date ('01/10/1998',  
'dd/mm/yyyy'), 'Bobeh', Gg198, '123456789')
```

```
3 into pegawai_012345 (id_pegawai, nama_depan,  
nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp)  
values (2, 'Budi', 'Subakti', to_date ('25/01/1997') dd/mm/yyyy',  
'madura', Gg71, '234567891')
```

```
4 into pegawai_012345 (id_pegawai, nama_depan, nama_  
belakang, tanggal_lahir, alamat, kode_pos, no_telp)  
values (3, 'Kukti', 'Sebakti', to_date ('08/01/1995', 'dd/mm/yyyy'))
```



Tugas Pertanyaan

'Sidoarjo', 6916, '345678912')

5 into pegawai_012345 (id_pegawai, nama_depan, nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp) values (4, 'Rini', 'Xrin', to_date('06/11/1998', 'dd/mm/yyyy'), 'Kungkut', 6971, '456789123')

6 into pegawai_012345 (id_pegawai, nama_depan, nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp) values (5, 'Beni', 'Pratama', to_date('31/05/1998', 'dd/mm/yyyy'), 'Sureolilo', 6901, '567891234')

7 select 1 from dual;

- Tabel Pelanggan

SQL > insert all

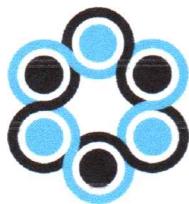
2 into pelanggan_012345 (id_pelanggan, nama_depan, nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp) values (1, 'Tomy', 'Taulana', to_date('31/10/1998', 'dd/mm/yyyy'), 'Lombok', 6919, '4234567891')

3 into pelanggan_012345 (id_pelanggan, nama_depan, nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp) values (2, 'Iudah', 'Nurain', to_date('28/01/1999', 'dd/mm/yyyy'), 'Madura', 6971, '234567891')

4 into pelanggan_012345 (id_pelanggan, nama_depan, nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp) values (3, 'Abdul', 'Falah', to_date('08/01/1994', 'dd/mm/yyyy'), 'Sidoarjo', 6916, '345678912')

5 into pelanggan_012345 (id_pelanggan, nama_depan, nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp) values (4, 'Izeno', 'Ubaridillah', to_date('06/11/1998', 'dd/mm/yyyy'), 'Jupiter', 6911, '456789123')

6 into pelanggan_012345 (id_pelanggan, nama_depan, nama_belakang, tanggal_lahir, alamat, kode_pos, no_telp) values (5, 'Anizal', 'Suhartono', to_date('31/05/1998', 'dd/mm/yyyy'), 'Pluto', 6901, '567891234')



Tugas Pertanyaan

- Tabel Kurir

SQL > insert all

2 into kurir_012345 (id_kurir, nama_perusahaan, no_telp) values
(1, 'Indomart', '123456789')

3 into kurir_012345 (id_kurir, nama_perusahaan, no_telp) values
(2, 'Alfamart', '234567891')

4 into kurir_012345 (id_kurir, nama_perusahaan, no_telp) values
(3, 'Giant', '345678912')

5 into kurir_012345 (id_kurir, nama_perusahaan, no_telp) values
(4, 'Transmart', '456789123')

6 into kurir_012345 (id_kurir, nama_perusahaan, no_telp) values
(5, 'Supernindo', '567891234')

7 select * from dual;

- Tabel Pemasok

SQL > insert all

2 into pemasok_012345 (id_pemasok, nama_perusahaan, alamat,
kode_pos, no_telp) values (1, 'DistroIndomart', 'Lambek', 6919, '123456789')

3 into pemasok_012345 (id_pemasok, nama_perusahaan, alamat, kode_pos,
no_telp) values (2, 'Distro Alfamart', 'Madura', 6921, '234567891')

4 into pemasok_012345 (id_pemasok, nama_perusahaan, alamat, kode_pos,
no_telp) values (3, 'Distro Giant', 'Sidoarjo', 6916, '345678912')

5 into pemasok_012345 (id_pemasok, nama_perusahaan, alamat, kode_pos,
no_telp) values (4, 'Distro Transmart', 'Jupiter', 6911, '456789123')

6 into pemasok_012345 (id_pemasok, nama_perusahaan, alamat,
kode_pos, no_telp) values (5, 'Distro Superindo', 'Pluto', 6901, '567891234')

7 select * from dual;

- Tabel Kategori

SQL > insert into kategori_012345 (id_kategori, nama_kategori) values
(1, 'Makanan');

1. row created.



Tugas Pertanyaan

SQL > insert into kategori_02345(id_kategori, nama_kategori) values (2, 'Minuman');
1 row created.

SQL > insert into kategori_02345(id_kategori, nama_kategori) values (3, 'Buku-Baju');
1 row created.

SQL > insert into kategori_02345(id_kategori, nama_kategori) values (4, 'Elektronik');
1 row created.

SQL > insert into kategori_02345(id_kategori, nama_kategori) values (5, 'Baju');
1 row created.

- Tabel Produk

SQL > insert all

2 into produk_02345(id_produk, id_pemasok, id_kategori, nama_produk, harga_satuan, stok_produk) values (1, 1, 1, 'Indomie', '2500', '50');

3 into produk_02345(id_produk, id_pemasok, id_kategori, nama_produk, harga_satuan, stok_produk) values (2, 2, 2, 'Marimas', '2000', '40');

4 into produk_02345(id_produk, id_pemasok, id_kategori, nama_produk, harga_rutton, stok_produk) values (2, 3, 3, 'Jeruk', '8000', '100');

5 into produk_02345(id_produk, id_pemasok, id_kategori, nama_produk, harga_satuan, stok_produk) values (4, 4, 4, 'Samsung', '8000000', '30');

6 into produk_02345(id_produk, id_pemasok, id_kategori, nama_produk, harga_satuan, stok_produk) values (11, 5, 'Supreme', '38000', '95')

7 select * from dual;

- Tabel Pemesanan

SQL > insert all

2 into pemesanan_02345(id_pemesanan, id_pelanegan, id_pejawarai, id_kurir, tanggal_pemesanan, tanggal_pengiriman, alamat_pengiriman, harga_total) values (1, 1, 1, 1, to_date('01/01/2017', 'dd/mm/yyyy'), to_date('02/01/2017', 'dd/mm/yyyy'), 'Bebek', '6000');



Tugas Pertanyaan

3 into pemesanan_0r3sy5 (id_pemesanan, id_pelanggan, id_pegawai, id_kurir,

tanggal_pemesanan, tanggal_pengiriman, alamat_pengiriman, harga_total)

values (2, 2, 2, to_date('02/02/2017', 'dd/mm/yyyy'), to_date(

'03/02/2017', 'dd/mm/yyyy'), 'Kungku', '50000')

4 into pemesanan_0r3sy5 (id_pemesanan, id_pelanggan, id_pegawai, id_kurir,

tanggal_pemesanan, tanggal_pengiriman, alamat_pengiriman, harga_total)

values (3, 3, 3, to_date('03/03/2017', 'dd/mm/yyyy'), to_date(

'04/03/2017', 'dd/mm/yyyy'), 'Sukelilo', '55000')

5 into pemesanan_0r3sy5 (id_pemesanan, id_pelanggan, id_pegawai, id_kurir,

tanggal_pemesanan, tanggal_pengiriman, alamat_pengiriman, harga_total)

values (4, 4, 4, to_date('04/04/2017', 'dd/mm/yyyy'), to_date(

'05/04/2017', 'dd/mm/yyyy'), 'Sukelilo', '100000')

6 into pemesanan_0r3sy5 (id_pemesanan, id_pelanggan, id_pegawai, id_kurir,

tanggal_pemesanan, tanggal_pengiriman, alamat_pengiriman, harga_total)

values (5, 5, 5, to_date('05/05/2017', 'dd/mm/yyyy'), to_date(

'06/05/2017', 'dd/mm/yyyy'), 'Sukelilo', '200000')

7 select 1 from dual ;

- Tabel Detail_Pemesanan

SQL > insert all

2 into detail_pemesanan_0r3sy5 (id_produk, id_pemesanan, jumlah, diskon) values ('1', '1', '1', '10')

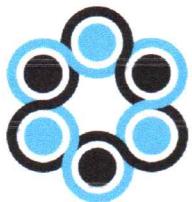
3 into detail_pemesanan_0r3sy5 (id_produk, id_pemesanan, jumlah, diskon) values ('2', '2', '2', '20')

4 into detail_pemesanan_0r3sy5 (id_produk, id_pemesanan, jumlah, diskon) values ('3', '3', '3', '30')

5 into detail_pemesanan_0r3sy5 (id_produk, id_pemesanan, jumlah, diskon) values ('4', '4', '4', '40')

6 into detail_pemesanan_0r3sy5 (id_produk, id_pemesanan, jumlah, diskon) values ('5', '5', '5', '50')

7 select 1 from dual ;



Tugas Pertanyaan

2. Updatelah dua record nama depan dan nama belakang pada tabel Pegawai!

- Tampilkan tabel yang ingin diupdate terlebih dahulu

SQL > SELECT * from pegawai-012345;

ID_PEGAWAI	NAMA_DEPAN	NAMA_BELAKANG	TANGGAL_ALAMAT	Kelulus	NR_TELP
1	Aulia	Mendian	01-OCT-98 Bandung	G998	(224)12345
2	Kim	Angga	25-JAN-97 Medan	G971	234-567891
3	Bakti	Seksanti	08-JAN-95 Balurjo	G916	345-678912
4	Rini	Arini	06-MAR-98 Rengat	G798	456-789123
5	Beni	Pratama	31-MAY-98 Sukabumi	G801	567891234

- Update tabel yang dinginkan

SQL > UPDATE pegawai-012345

2 SET nama_depan = 'Kim';

3 nama_belakang = 'Jiso';

4 where id_pegawai = 1;

1 row updated.

SQL >

SQL > UPDATE pegawai-012345

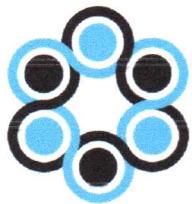
2 SET nama_depan = 'Lee';

3 nama_belakang = 'Kurniawan'

4 where id_pegawai = 2;

1 row updated.

- Cek ulang pada tabel yang telah diupdate



Tugas Pertanyaan

SQL > SELECT * from pegawai_cr2345;

ID_PEGAWAI	NAMA	BEDAN_NAMA	BELAKANG	TANGGALL_L	JAMATHT	KODE_KOS	MULTELP
1	Kim	Jisoo		01-OCT-98	Bebek	69198	123456789
2	Lee	Kurniawan		20-JAN-97	Madura	6921	234567891
3	Bakti	Sesakti		01-JAN-95	Peleburjo	6916	345678912
4	Kim	Aini		06-Nov-98	Kengkut	67911	456789123
5	Beni	Pratama		31-Nov-96	Sukedile	6501	567891234

3. Hapus dua record pada tabel detail_pemesanan !

- Tampilkan tabel yang ingin dihapus

SQL > SELECT * from detail_pemesanan_cr2345;

ID_PRODUK	ID_PEMESANAN	JUMLAH	DISKON
1	1	1	10
2	2	2	20
3	3	3	30
4	4	4	40
5	5	5	50

- Hapus tabel yang dituju

SQL > DELETE FROM detail_pemesanan_cr2345

2 where id_produk = 4 ;

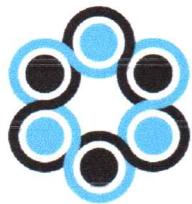
1 row deleted.

SQL >

SQL > DELETE FROM detail_pemesanan_cr2345

2 where id_produk = 5 ;

1 row deleted.



Tugas Pertanyaan

- Cek ulang tabel yang telah dihapus

SQL > SELECT * from detail_pemesanan - 012345;

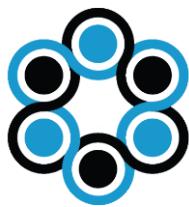
ID_PEMERIKSA	ID_PEMESANAN	JUMLAH	DISKON
1	1	1	10
2	2	2	20
3	3	3	30

4. Buatlah query dimana menampilkan kode_pos, alamat, no_telp, tanggal_lahir dari setiap pegawai yang nama depannya memiliki huruf a (semua) dan mempunyai nama belakang e (semua) dengan id_pegawai kurang dari 5 ?

SQL > SELECT kode_pos, alamat, no_telp, tanggal_lahir from
pegawai - 012345
2 where nama_depan LIKE '%a%' AND nama_belakang LIKE
'%e%' AND id_pegawai < 5;

KODE_POS ALAMAT NO-TELP TANGGAL-L

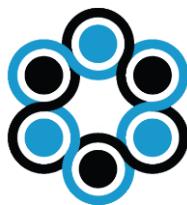
6916 Sidoarjo 345678912 08-JAN-95



Tugas Praktikum

Soal Praktikum

1. Insert data pada setiap tabel yang telah Anda buat :
 - a. Single insert dengan 2 data
 - b. Multiple insert dengan 3 data
 - c. Terapkan insert dengan sequence yang telah Anda buat di Modul 1
2. Update 5 data pada tabel yang berbeda :
 - a. Semua data pada suatu kolom pada tabel
 - b. Menerapkan where clause
 - c. Menerapkan like
 - d. Menerapkan AND, OR dan NOT minimal 2 operator pada satu baris query
3. Delete minimal 3 data pada salah satu tabel dengan menerapkan 1 klausa dan 2 operator pada satu baris query yang berbeda pada setiap data
4. Aturlah transaksi dengan menerapkan :
 - a. Save Point
 - b. Commit
 - c. Rollback
5. Terapkan select dengan menerapkan :
 - a. Order By
 - b. Group By



Tugas Praktikum

Langkah Pertama

Insert 5 data pada table **Employee** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO employee (id_employee,
employee_name, employee_addr, gender,
phone_no, email) values (101, 'Muchlas',
'Dusun Duran Sedati', 'L', '8124186',
'muchlas@gmail.com');

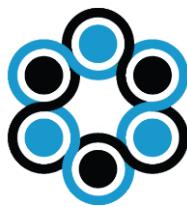
INSERT INTO employee (id_employee,
employee_name, employee_addr, gender,
phone_no, email) values (102, 'Sterling',
'Surabaya Sukolilo', 'L', '8225045',
'sterling@gmail.com');

INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (103, 'Naura', 'Rungkut Surbaya', 'P',
'8774501', 'naura@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (104, 'Alya', 'WR Supratman Surabaya',
'P', '85643801', 'alya@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (105, 'John', 'Gedangan Sidoarjo',
'L', '811345716', 'john@gmail.com')
SELECT 1 FROM dual;)
```

Screenshot

```
SQL> INSERT INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(101, 'Muchlas', 'Dusun Duran Sedati', 'L', '8124186', 'muchlas@gmail.com');
1 row created.

SQL> INSERT INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(102, 'Sterling', 'Surabaya Sukolilo', 'L', '8225045', 'sterling@gmail.com');
1 row created.
```



Tugas Praktikum

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (103,
Naura', 'Rungkut Surbaya', 'P', '8774501', 'naura@gmail.com')
  3  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (104,
'Alya', 'WR Supratman Surabaya', 'P', '85643801', 'alya@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (105,
'John', 'Gedangan Sidoarjo', 'L', '811345716', 'john@gmail.com')
  5  SELECT 1 FROM dual;

3 rows created.

SQL> SELECT * FROM employee;
ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----101 Muchlas
Dusun Duran Sedati
L 8124186 muchlas@gmail.com

102 Sterling
Surabaya Sukolilo
L 8225045 sterling@gmail.com

ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----103 Naura
Rungkut Surbaya
P 8774501 naura@gmail.com

104 Alya
WR Supratman Surabaya

ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----P 85643801 alya@gmail.com

105 John
Gedangan Sidoarjo
L 811345716 john@gmail.com

SQL>
```

Analisa

Gambar #1

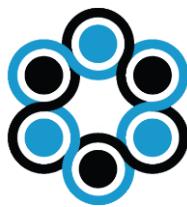
Insert dilakukan pada table **Employee** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INERT INTO** dengan data sebagai berikut:

- 101, 'Muchlas', 'Dusun Duran Sedati', 'L', '8124186', 'muchlas@gmail.com'
- 102, 'Sterling', 'Surabaya Sukolilo', 'L', '8225045', 'sterling@gmail.com'

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.

Gambar #2

Insert dilakukan pada table **Employee** dengan 3 baris selanjutnya menggunakan Multiple Insert menggunakan perintah **INSERT ALL** dengan data sebagai berikut:



Tugas Praktikum

- 103, 'Naura', 'Rungkut Surbaya', 'P', '8774501', 'naura@gmail.com'
- 104, 'Alya', 'WR Supratman Surabaya', 'P', '85643801', 'alya@gmail.com'
- 105, 'John', 'Gedangan Sidoarjo', 'L', '811345716', 'john@gmail.com'

Dengan ditambah penggunaan klausa **SELECT 1 FROM dual** pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Employee**. Sesuai query masing – masing (baik single insert maupun multiple insert).

Langkah Kedua

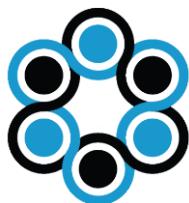
Insert 5 data pada table **Client** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (11, 'Hexamedika', 'Rungkut Surabaya',
'admin@hexa.co.id', '14041');

INSERT INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (12, 'Ekalya Vessel', 'Darmo
Surabaya', 'hrd@ekalya.co.id', '14042');

INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (13, 'Avodamitra', 'Semampir
Sidoarjo', 'hrd@avoda.co.id', '14043')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (14, 'Golden Union', 'Jabon Sidoarjo',
'hr.ga@golden.co.id', '14044')
```



Tugas Praktikum

```
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (15, 'Orela Shipyard', 'Pangkah
Gresik', 'support@orela.co.id', '14045')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1,
1, 'Hexamedika', 'Rungkut Surabaya', 'admin@hexa.co.id', '14041');
1 row created.

SQL> INSERT INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
2, 'Ekalya Vessel', 'Darmo Surabaya', 'hrd@ekalya.co.id', '14042');
1 row created.

SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (13, 'Avo
damitra', 'Semampir Sidoarjo', 'hrd@avoda.co.id', '14043')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (14, 'Gol
den Union', 'Jabon Sidoarjo', 'hr.ga@golden.co.id', '14044')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (15, 'Ore
la Shipyard', 'Pangkah Gresik', 'support@orela.co.id', '14045')
  5  SELECT 1 FROM dual;
```

```
SQL> select * from client;
      ID_CLIENT CLIENT_NAME          CLIENT_ADDR           CLIENT_EMAIL
      CLIENT_PHONE_NO
      -----
      11 Hexamedika          Rungkut Surabaya    admin@hexa.co.id
      14041
      12 Ekalya Vessel        Darmo Surabaya    hrd@ekalya.co.id
      14042
      13 Avodamitra          Semampir Sidoarjo  hrd@avoda.co.id
      14043
      14 Golden Union         Jabon Sidoarjo   hr.ga@golden.co.id
      14044
      15 Orela Shipyard       Pangkah Gresik   support@orela.co.id
      14045
```

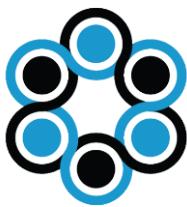
Analisa :

Gambar #1

Insert dilakukan pada table **Client** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- 11, 'Hexamedika', 'Rungkut Surabaya', 'admin@hexa.co.id', '14041'
- 12, 'Ekalya Vessel', 'Darmo Surabaya', 'hrd@ekalya.co.id', '14042'

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.



Tugas Praktikum

Gambar #2

Insert dilakukan pada table **Client** dengan 3 baris selanjutnya menggunakan Multiple Insert menggunakan perintah **INSERT ALL** dengan data sebagai berikut:

- 13, 'Avodamitra', 'Semampir Sidoarjo', 'hrd@avoda.co.id', '14043'
- 14, 'Golden Union', 'Jabon Sidoarjo', 'hr.ga@golden.co.id', '14044'
- 15, 'Orela Shipyard', 'Pangkah Gresik', 'support@orela.co.id', '14045'

Dengan ditambah penggunaan klausa `SELECT 1 FROM dual` pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Client**. Sesuai query masing – masing (baik single insert maupun multiple insert). Tidak ada error saat entry data maupun saat menampilkan data.

Langkah Ketiga

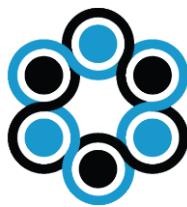
Insert 5 data pada table **Job** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO job (id_job, id_client, job_name,
category, description) values (1101, 11,
'Network Administrator', 'IT', 'Manage and
monitoring network');

INSERT INTO job (id_job, id_client, job_name,
category, description) values (1102, 12,
'Front-End Developer', 'IT', 'Design and
develop UI Website');
```

```
INSERT ALL
INTO job (id_job, id_client, job_name,
category, description) values (1103, 13, 'RAC
Engineer', 'Engineer', 'Install and maintain')
```



Tugas Praktikum

```
INTO job (id_job, id_client, job_name,
category, description) values (1104, 14,
'Accounting Staff', 'Finance', 'Reviewing
financial statement')
INTO job (id_job, id_client, job_name,
category, description) values (1105, 15,
'QA/QC Vessel', 'Engineer', 'Inspect and test
materials')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT INTO job (id_job, id_client, job_name, category, description) values (1101, 11, 'Network A
dministrator', 'IT', 'Manage and monitoring network');
1 row created.

SQL> INSERT INTO job (id_job, id_client, job_name, category, description) values (1102, 12, 'Front-End
Developer', 'IT', 'Design and develop UI Website');
1 row created.

SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1103, 13, 'RAC Engineer', '
Engineer', 'Install and maintain cooling system')
  3  INTO job (id_job, id_client, job_name, category, description) values (1104, 14, 'Accounting Staff
', 'Finance', 'Reviewing financial statement')
  4  INTO job (id_job, id_client, job_name, category, description) values (1105, 15, 'QA/QC Vessel', '
Engineer', 'Inspect and test materials')
  5  SELECT 1 FROM dual;
3 rows created.

SQL> select * from job;
   ID_JOB ID_CLIENT JOB_NAME          CATEGORY
DESCRIPTION
----- -----
 1101      11 Network Administrator        IT
           Manage and monitoring network
 1102      12 Front-End Developer        IT
           Design and develop UI Website
 1103      13 RAC Engineer             Engineer
           Install and maintain cooling system
 1104      14 Accounting Staff          Finance
           Reviewing financial statement
 1105      15 QA/QC Vessel            Engineer
           Inspect and test materials
```

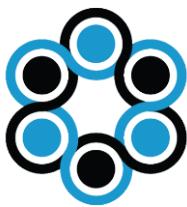
Analisa :

Gambar #1

Insert dilakukan pada table **Job** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- 1101, 11, 'Network Administrator', 'IT', 'Manage and monitoring network'
- 1102, 12, 'Front-End Developer', 'IT', 'Design and develop UI Website'

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.



Tugas Praktikum

Gambar #2

Insert dilakukan pada table **Job** dengan 3 baris selanjutnya menggunakan Multiple Insert menggunakan perintah **INSERT ALL** dengan data sebagai berikut:

- 1103, 13, 'RAC Engineer', 'Engineer', 'Install and maintain cooling system'
- 1104, 14, 'Accounting Staff', 'Finance', 'Reviewing financial statement'
- 1105, 15, 'QA/QC Vessel', 'Engineer', 'Inspect and test materials'

Dengan ditambah penggunaan klausa `SELECT 1 FROM dual` pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Job**. Sesuai query masing – masing (baik single insert maupun multiple insert). Tidak ada error saat entry data maupun saat menampilkan data.

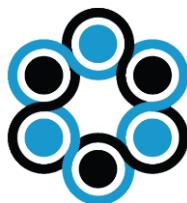
Langkah Keempat

Insert 5 data pada table **Project** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1101, 101,
to_date('01/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy'));

INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1102, 102,
to_date('02/05/2020', 'dd/mm/yyyy'),
to_date('10/05/2020', 'dd/mm/yyyy'));
```



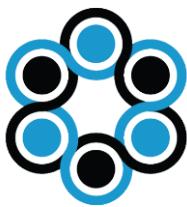
Tugas Praktikum

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1103, 103,
to_date('03/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy'))  
  
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1104, 104,
to_date('04/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy'))  
  
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1105, 103,
to_date('05/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy'))
```

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1101, 101, to_date('01/05/2020', 'dd/mm/yyyy'), to_date('20/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1102, 102, to_date('02/05/2020', 'dd/mm/yyyy'), to_date('10/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1103, 103, to_date('03/05/2020', 'dd/mm/yyyy'), to_date('30/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1104, 104, to_date('04/05/2020', 'dd/mm/yyyy'), to_date('20/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.ne
xtval, 1105, 105, to_date('05/05/2020', 'dd/mm/yyyy'), to_date('15/05/2020', 'dd/mm/yyyy'));
1 row created.  
SQL>
```

```
SQL> select * from project;  
ID_PROJECT      ID_JOB  ID_EMPLOYEE START_DATE END_DATE  
-----  
      3          1101      101 01-MAY-20 20-MAY-20  
      4          1102      102 02-MAY-20 10-MAY-20  
      6          1103      103 03-MAY-20 30-MAY-20  
      7          1104      104 04-MAY-20 20-MAY-20  
      8          1105      105 05-MAY-20 15-MAY-20
```



Tugas Praktikum

Analisa :

Insert dilakukan pada table **Project** dengan 2 baris pertama menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- id_project.nextval, 1101, 101, to_date('01/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy')
- id_project.nextval, 1102, 102, to_date('02/05/2020', 'dd/mm/yyyy'),
to_date('10/05/2020', 'dd/mm/yyyy')

Dengan menggunakan perintah **INSERT INTO** serta menyertakan *col_list (column list)* data yang dientry harus dieksekusi satu per satu.

Gambar #2

Insert dilakukan pada table **Project** dengan 3 baris selanjutnya masih menggunakan Single Insert menggunakan perintah **INSERT INTO** dengan data sebagai berikut:

- id_project.nextval, 1103, 103, to_date('03/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy')
- id_project.nextval, 1104, 104, to_date('04/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy')
- id_project.nextval, 1105, 103, to_date('05/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy')

Dengan ditambah penggunaan klausa **SELECT 1 FROM dual** pada akhir query **INSERT ALL**. Masih sama dengan query sebelumnya, dengan menyertakan *col_list (column list)* data yang dientry bisa dieksekusi dalam 1 line query tanpa harus dieksekusi satu per satu.

Gambar #3

Hasil dari data yang sudah dientry atau dimasukkan ke dalam table **Project**. Sesuai query masing – masing (baik single insert maupun multiple insert). Tidak ada error saat entry data maupun saat menampilkan data.



Tugas Praktikum

Langkah Kelima

Update data berupa **id_client**, **job_name**, **category** dan **description** pada table **Job** dengan menggunakan query **UPDATE** dasar atau basic.

Query

```
UPDATE job
SET id_client = 15,
job_name = 'QA Vessel',
category = 'Vessel Engineer',
description = 'quality assessment'
WHERE id_job = 1103
```

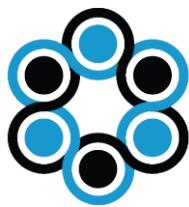
Screenshot

```
SQL> UPDATE job
  2  SET id_client = 15,
  3  job_name = 'QA Vessel',
  4  category = 'Vessel Engineer',
  5  description = 'quality assessment'
  6 WHERE id_job = 1103;
1 row updated.

SQL> SELECT * FROM job;
      ID_JOB ID_CLIENT JOB_NAME          CATEGORY           DESCRIPTION
-----  -----  -----
        1101      11 Network Engineer      IT Support      Maintain and
monitoring network
        1102      12 Network Engineer      IT Support      Maintain, ins
tall, and monitoring network
        1103      15 QA Vessel          Vessel Engineer  quality asses
sment
        1104      14 Accounting Staff     Finance        Reviewing fin
ancial statement
        1105      15 QA/QC Vessel        Engineer       Inspect and t
est materials
SQL>
```

Analisa :

Melakukan update pada table **Job** dengan data yang di update adalah **id_client**, **job_name**, **category** dan **description** dengan kondisi dimana data pada **id_job** memiliki nilai atau berisikan data **1103**. Selain **id_job** yang dientry-kan pada klausa **WHERE** tidak akan mengeksekusi query **UPDATE**.



Tugas Praktikum

Langkah Keenam

Update data berupa **id_client**, **job_name**, **category** dan **description** pada table **Client** dengan menggunakan query **UPDATE** dan klausula **WHERE**.

Query

```
UPDATE client
SET client_name = 'Hexa-medika Corp',
client_addr = 'Rungkut Industri Surabaya'
WHERE client_email LIKE 'admin@hexa%' AND
client_name = 'Hexamedika' AND id_client <=
13;
```

Screenshot

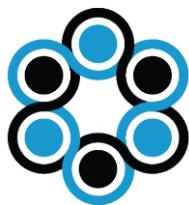
```
SQL> UPDATE client
2 SET client_name = 'Hexa-medika Corp',
3 client_addr = 'Rungkut Industri Surabaya'
4 WHERE client_email LIKE 'admin@hexa%' AND client_name = 'Hexamedika' AND id_client <= 13;
1 row updated.

SQL> SELECT * FROM client;
ID_CLIENT CLIENT_NAME           CLIENT_ADDR          CLIENT_PHONE_NO
-----  -----
11 Hexa-medika Corp      Rungkut Industri Surabaya   14041
12 Ekalya Vessel        Darmo Surabaya          14042
13 CV Avodamitra       Semampir Sidoarjo      14043
14 Golden Union         Jaban Sidoarjo        14044
15 Orela Shipyard       Pangkah Gresik        14045

SQL>
```

Analisa :

Melakukan update pada table **Client** dengan data yang di update adalah **client_name** dan **client_addr** dengan menggunakan klausula **WHERE** untuk kondisi dimana data pada **client_email** memiliki karakteristik kata **admin@hexa** di awal kalimat **DAN** **client_name = Hexamedika DAN id_client <= 13**



Tugas Praktikum

Langkah Ketujuh

Update data berupa **employee_name** dan **employee_addr** pada table **Employee** dengan menerapkan klausula **WHERE** dan klausula **LIKE**.

Query

```
UPDATE employee
SET employee_name = 'Naurasari',
employee_addr = 'Taman Bungkul Surabaya'
WHERE email LIKE 'naura%' AND employee_name
LIKE 'Naura%' AND id_employee = 103;
```

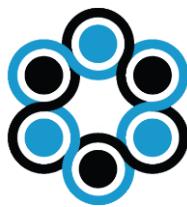
Screenshot

```
SQL> UPDATE employee
  2 SET employee_name = 'Naurasari',
  3 employee_addr = 'Taman Bungkul Surabaya'
  4 WHERE email LIKE 'naura%' AND employee_name LIKE 'Naura%' AND id_employee = 103;
1 row updated.

SQL> select * from employee;
ID_EMPLOYEE EMPLOYEE_NAME          EMPLOYEE_ADDR          GEN PHONE_NO EMAIL
-----  -----
          101 Muchlas           Dusun Duran Sedati      L   8124186 muchlas@gmai
mail.com
          102 Sterling          Surabaya Sukolilo       L   8225045 sterling@gmai
gmail.com
          103 Naurasari         Taman Bungkul Surabaya P   8774501 naura@gmai
il.com
          104 Alya              WR Supratman Surabaya P   85643801 alya@gmai
l.com
          105 John              Gedangan Sidoarjo     L   811345716 john@gmai
l.com
SQL>
```

Analisa :

Melakukan update pada table **Employee** dengan data yang di update adalah **employee_name** dan **employee_addr** dengan menerapkan klausula **WHERE** dan klausula **LIKE** untuk kondisi dimana data pada kolom email memiliki karakteristik kata **naura** di awal kalimat **DAN** **employee_name** memiliki karakteristik **Naura** di awal kalimat **DAN** **id_employee = 103**.



Tugas Praktikum

Langkah Kedelapan

Update data berupa **client_name** dan **client_email** pada table **Client** dengan menerapkan klausa **WHERE**, klausa **LIKE**, beserta penerapan operator **AND** atau **OR** atau **NOT**.

Query

```
UPDATE client
SET client_name = 'CV Avodamitra',
client_email = 'hrd@avodamitra.co.id'
WHERE client_name LIKE 'Avoda%' OR
client_email LIKE 'hrd@avo%' AND id_client
< 15;
```

Screenshot

```
SQL> UPDATE client
2 SET client_name = 'CV Avodamitra',
3 client_email = 'hrd@avodamitra.co.id'
4 WHERE client_name LIKE 'Avoda%' OR client_email LIKE 'hrd@avo%' AND id_client < 15;

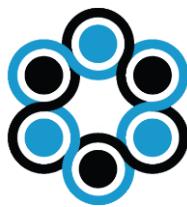
1 row updated.

SQL> SELECT * FROM client;
ID_CLIENT CLIENT_NAME           CLIENT_ADDR          CLIENT_EMAIL
CLIENT_PHONE_NO
-----
11 Hexamedika                 Rungkut Surabaya    admin@hexa.co.id
14041
12 Ekalya Vessel               Darmo Surabaya     hrd@ekalya.co.id
14042
13 CV Avodamitra              Semampir Sidoarjo   hrd@avodamitra.co.id
14043
14 Golden Union                Jabon Sidoarjo    hr.ga@golden.co.id
14044
15 Orela Shipyard              Pangkah Gresik    support@orela.co.id
14045

SQL>
```

Analisa :

Melakukan update pada table **Client** dengan data yang di update adalah **client_name** dan **client_email** dengan menerapkan klausa **WHERE** dan klausa **LIKE** untuk kondisi dimana data pada kolom **client_name** memiliki karakteristik kata **Avoda** di awal kalimat **ATAU** **client_email** memiliki karakteristik **hrd@avo%** di awal kalimat **DAN** **id_client < 15**.



Tugas Praktikum

Langkah Kesembilan

Update data berupa `start_date` dan `id_employee` pada table **Project** dengan menerapkan klausa **WHERE**, beserta penerapan operator **AND** atau **OR** atau **NOT**.

Query

```
UPDATE project
SET start_date = to_date('07/05/2020',
'dd/mm/yyyy'),
id_employee = 103
WHERE id_project <= 8 AND start_date =
to_date('05/05/2020', 'dd/mm/yyyy') AND
end_date >= to_date('15/05/2020',
'dd/mm/yyyy');
```

Screenshot

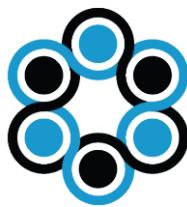
```
SQL> UPDATE project
  2  SET start_date = to_date('07/05/2020', 'dd/mm/yyyy'),
  3  id_employee = 103
  4 WHERE id_project <= 8 AND start_date = to_date('05/05/2020', 'dd/mm/yyyy') AND end_
date >= to_date('15/05/2020', 'dd/mm/yyyy');

1 row updated.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----  -----
      3        1101      101 01-MAY-20 20-MAY-20
      4        1102      102 02-MAY-20 18-MAY-20
      6        1103      103 03-MAY-20 30-MAY-20
      7        1104      104 04-MAY-20 20-MAY-20
      8        1105      103 07-MAY-20 15-MAY-20
```

Analisa :

Melakukan update pada table **Project** dengan data yang di update adalah `start_date` dan `id_employee` dengan menerapkan klausa **WHERE** untuk kondisi dimana data pada kolom `id_project` memiliki kondisi kurang dari sama dengan **8 DAN start_date = 05/05/2020 DAN end_date = 15/05/2020**.



Tugas Praktikum

Langkah Kesepuluh

Delete 3 data pada table **Project** dengan penggunaan kondisi yang berbeda – beda.

Query

```
DELETE FROM project
WHERE start_date = to_date('01/05/2020',
'dd/mm/yyyy') AND end_date =
to_date('20/05/2020', 'dd/mm/yyyy') AND
id_project < 8;

DELETE FROM project
WHERE end_date = to_date('20/05/2020',
'dd/mm/yyyy') AND start_date >
to_date('03/05/2020', 'dd/mm/yyyy') OR
id_project = 7;

DELETE FROM project
WHERE start_date BETWEEN
to_date('02/05/2020', 'dd/mm/yyyy') AND
to_date('06/05/2020', 'dd/mm/yyyy') AND
id_project > 4;
```

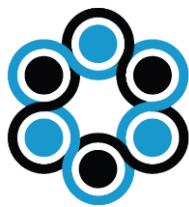
Screenshot

```
SQL> DELETE FROM project
  2 WHERE start_date = to_date('01/05/2020', 'dd/mm/yyyy') AND end_date = to_date('20/05/2020', 'dd/mm/yyyy') AND id_project < 8;
1 row deleted.

SQL> DELETE FROM project
  2 WHERE end_date = to_date('20/05/2020', 'dd/mm/yyyy') AND start_date > to_date('03/05/2020', 'dd/mm/yyyy') OR id_project = 7;
1 row deleted.

SQL> DELETE FROM project
  2 WHERE start_date BETWEEN to_date('02/05/2020', 'dd/mm/yyyy') AND to_date('06/05/2020', 'dd/mm/yyyy') AND id_project > 4;
1 row deleted.

SQL>
```



Tugas Praktikum

```
SQL> SELECT * FROM project;  
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE  
-----  
        4          1102      102 02-MAY-20 10-MAY-20  
        8          1105      103 07-MAY-20 15-MAY-20  
SQL>
```

Analisa :

Gambar #1

Melakukan delete dengan menggunakan query **DELETE** dan menerapkan klausa **WHERE** dimana kondisi nya adalah **start_date = 01/05/2020 DAN end_date = 20/05/2020 DAN id_project < 8.**

Gambar #2

Melakukan delete dengan menggunakan query **DELETE** dan menerapkan klausa **WHERE** dimana kondisi nya adalah **end_date = 20/05/2020 DAN start_date lebih dari 03/05/2020 DAN id_project = 7.**

Gambar #3

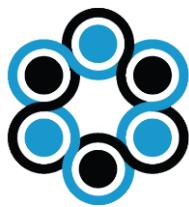
Melakukan delete dengan menggunakan query **DELETE** dan menerapkan klausa **WHERE** dimana kondisi nya adalah **start_date BETWEEN 02/05/2020 DAN 06/05/2020 DAN id_project lebih dari 4.**

Langkah Kesebelas

Melakukan perintah **SAVEPOINT** untuk menyimpan **CHECKPOINT** dimana data terakhir disimpan.

Query

```
SAVEPOINT del_project;
```



Tugas Praktikum

Screenshot

```
SQL> SELECT * FROM project;

ID_PROJECT    ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----  -----  -----  -----  -----
        4      1102      102 02-MAY-20 10-MAY-20
        8      1105      103 07-MAY-20 15-MAY-20

SQL> SAVEPOINT del_project;
Savepoint created.

SQL>
```

Analisa :

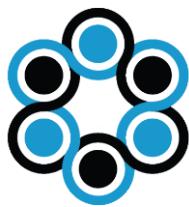
Dapat dilihat pada gambar di atas, dengan menggunakan klaus **SELECT** untuk melihat data yang up to date pada table **Project** untuk selanjutnya dibuatkan **SAVEPOINT** dengan nama **del_project**. Penamaan **SAVEPOINT** sangatlah berpengaruh, untuk memudahkan kita mengingat apabila saat proses entry data ada kesalahan lalu kita ingin **ROLLBACK** ke **SAVEPOINT** yang terakhir kita buat.

Langkah Keduabelas

Melakukan perintah **ROLLBACK** untuk mengembalikan dimana data terakhir disimpan pada pembuatan **SAVEPOINT**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1104, 101,
to_date('10/05/2020', 'dd/mm/yyyy'),
to_date('22/05/2020', 'dd/mm/yyyy'))
ROLLBACK TO SAVEPOINT del_project;
```



Tugas Praktikum

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1104, 101, to_date('10/05/2020', 'dd/mm/yyyy'), to_date('22/05/2020
', 'dd/mm/yyyy'));
1 row created.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----      -----      -----      -----      -----
        4          1102        102 02-MAY-20 10-MAY-20
        8          1105        103 07-MAY-20 15-MAY-20
        9          1104        101 10-MAY-20 22-MAY-20

SQL> ROLLBACK TO SAVEPOINT del_project;
Rollback complete.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----      -----      -----      -----      -----
        4          1102        102 02-MAY-20 10-MAY-20
        8          1105        103 07-MAY-20 15-MAY-20

SQL>
```

Analisa :

Setelah melakukan **SAVEPOINT** pada langkah sebelumnya, pada langkah ini kita akan mencoba melakukan **ROLLBACK** data dengan percobaan entry data terlebih dahulu pada table **Project**. Setelah entry data pada table **Project**, otomatis data yang ada pada table tersebut akan bertambah. Dengan menggunakan query **ROLLBACK TO SAVEPOINT del_project**, kita bisa mengembalikan kondisi data terakhir saat **SAVEPOINT** dibuat.

Langkah Ketigabelas

Melakukan perintah **COMMIT** untuk menyimpan secara permanen dari perubahan data yang terakhir.

Query

```
COMMIT;
```



Tugas Praktikum

Screenshot

```
SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----
        4           1102      102 02-MAY-20 10-MAY-20
        8           1105      103 07-MAY-20 15-MAY-20
SQL> COMMIT;
Commit complete.
SQL>
```

Analisa :

Langkah yang selanjutnya adalah melakukan **COMMIT**. Setelah melakukan **ROLLBACK** pada table **Project** data akan kembali dimana **SAVEPOINT** dibuat. Ketika data tersebut sudah fix dan akan disimpan secara permanen tanpa ada perubahan data lagi, maka kita gunakan query **COMMIT** untuk menyimpan kondisi data terakhir secara permanen (biasanya dilakukan pada akhir entry / update data).

Langkah Keempatbelas

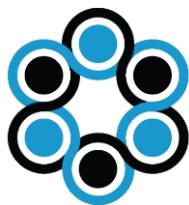
Melakukan perintah **SELECT** dengan menggunakan atau menerapkan kondisi **ORDER BY** pada table **Project**.

Query

```
SELECT * FROM project ORDER BY end_date;
```

Screenshot

```
SQL> SELECT * FROM project ORDER BY end_date;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----
        4           1102      102 02-MAY-20 10-MAY-20
        8           1105      103 07-MAY-20 15-MAY-20
```



Tugas Praktikum

Analisa :

Pada langkah ini, bisa dilihat pada gambar di atas dengan menggunakan klausu **SELECT** untuk menampilkan data pada table **Project** ditambah dengan query **ORDER BY** pada kolom `end_date` maka tampilan pada table **Project** akan diurutkan berdasarkan kolom `end_date`.

Langkah Kelimabelas

Melakukan perintah **SELECT** dengan menggunakan atau menerapkan kondisi **GROUP BY** pada table **Project**.

Query

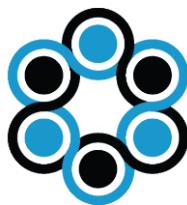
```
SELECT end_date, count(id_job) FROM project  
GROUP BY end_date;
```

Screenshot

```
SQL> SELECT end_date, count(id_job) FROM project GROUP BY end_date;  
END_DATE COUNT(ID_JOB)  
-----  
15-MAY-20      1  
22-MAY-20      1  
SQL>
```

Analisa :

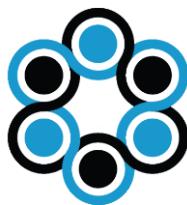
Langkah yang terakhir adalah menampilkan data pada table **Project** yang ditambahkan dengan query **GROUP BY** kolom `end_date`, maka data yang ada pada table **Project** akan menunjukkan pengelompokan data berdasarkan `end_date` dan juga terdapat fungsi agregat **COUNT** untuk menghitung berapa jumlah `id_job` pada tanggal tersebut.



Laporan Sementara

Soal Praktikum

1. Insert data pada setiap tabel yang telah Anda buat :
 - a. Single insert dengan 2 data
 - b. Multiple insert dengan 3 data
 - c. Terapkan insert dengan sequence yang telah Anda buat di Modul 1
2. Update 5 data pada tabel yang berbeda :
 - a. Semua data pada suatu kolom pada tabel
 - b. Menerapkan where clause
 - c. Menerapkan like
 - d. Menerapkan AND, OR dan NOT minimal 2 operator pada satu baris query
3. Delete minimal 3 data pada salah satu tabel dengan menerapkan 1 klausa dan 2 operator pada satu baris query yang berbeda pada setiap data
4. Aturlah transaksi dengan menerapkan :
 - a. Save Point
 - b. Commit
 - c. Rollback
5. Terapkan select dengan menerapkan :
 - a. Order By
 - b. Group By



Laporan Sementara

Langkah Pertama

Insert 5 data pada table **Employee** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO employee (id_employee,
employee_name, employee_addr, gender,
phone_no, email) values (101, 'Muchlas',
'Dusun Duran Sedati', 'L', '8124186',
'muchlas@gmail.com');

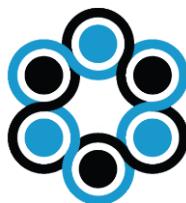
INSERT INTO employee (id_employee,
employee_name, employee_addr, gender,
phone_no, email) values (102, 'Sterling',
'Surabaya Sukolilo', 'L', '8225045',
'sterling@gmail.com');

INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (103, 'Naura', 'Rungkut Surbaya', 'P',
'8774501', 'naura@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (104, 'Alya', 'WR Supratman Surabaya',
'P', '85643801', 'alya@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (105, 'John', 'Gedangan Sidoarjo',
'L', '811345716', 'john@gmail.com')
SELECT 1 FROM dual;)
```

Screenshot

```
SQL> INSERT INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(101, 'Muchlas', 'Dusun Duran Sedati', 'L', '8124186', 'muchlas@gmail.com');
1 row created.

SQL> INSERT INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(102, 'Sterling', 'Surabaya Sukolilo', 'L', '8225045', 'sterling@gmail.com');
1 row created.
```



Laporan Sementara

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (103, 'Naura', 'Rungkut Surbaya', 'P', '8774501', 'naura@gmail.com')
  3  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (104, 'Alya', 'WR Supratman Surabaya', 'P', '85643801', 'alya@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values (105, 'John', 'Gedangan Sidoarjo', 'L', '811345716', 'john@gmail.com')
  5  SELECT 1 FROM dual;

3 rows created.

SQL> SELECT * FROM employee;
ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----101 Muchlas
Dusun Duran Sedati
L 8124186 muchlas@gmail.com

102 Sterling
Surabaya Sukolilo
L 8225045 sterling@gmail.com

ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----103 Naura
Rungkut Surbaya
P 8774501 naura@gmail.com

104 Alya
WR Supratman Surabaya

ID_EMPLOYEE EMPLOYEE_NAME
-----EMPLOYEE_ADDR
-----GEN PHONE_NO EMAIL
-----P 85643801 alya@gmail.com

105 John
Gedangan Sidoarjo
L 811345716 john@gmail.com

SQL>
```

Analisa

Gambar #1

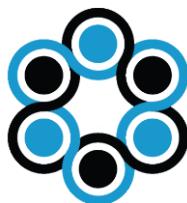
Insert dilakukan pada table **Employee** dengan 2 baris pertama menggunakan Single Input menggunakan perintah **INERT INTO**.

Gambar #2

Insert dilakukan pada table **Employee** dengan 3 baris selanjutnya menggunakan Multiple Input menggunakan perintah **INERT ALL** ditambah pada akhir baris menggunakan klausa **SELECT 1 FROM dual**;

Gambar #3

Hasil dari data yang sudah diinsert atau dimasukkan ke dalam table **Employee**.



Laporan Sementara

Langkah Kedua

Insert 5 data pada table **Client** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (11, 'Hexamedika', 'Rungkut Surabaya',
'admin@hexa.co.id', '14041');

INSERT INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (12, 'Ekalya Vessel', 'Darmo
Surabaya', 'hrd@ekalya.co.id', '14042');

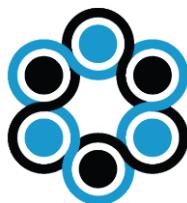
INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (13, 'Avodamitra', 'Semampir
Sidoarjo', 'hrd@avoda.co.id', '14043')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (14, 'Golden Union', 'Jabon Sidoarjo',
'hr.ga@golden.co.id', '14044')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (15, 'Orela Shipyard', 'Pangkah
Gresik', 'support@orela.co.id', '14045')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
1, 'Hexamedika', 'Rungkut Surabaya', 'admin@hexa.co.id', '14041');
1 row created.

SQL> INSERT INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
2, 'Ekalya Vessel', 'Darmo Surabaya', 'hrd@ekalya.co.id', '14042');
1 row created.

SQL> INSERT ALL
2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (13, 'Av
damitra', 'Semampir Sidoarjo', 'hrd@avoda.co.id', '14043')
3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (14, 'Gol
den Union', 'Jabon Sidoarjo', 'hr.ga@golden.co.id', '14044')
4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (15, 'Ore
la Shipyard', 'Pangkah Gresik', 'support@orela.co.id', '14045')
5  SELECT 1 FROM dual;
2 rows selected.
```



Laporan Sementara

ID_CLIENT	CLIENT_NAME	CLIENT_PHONE_NO	CLIENT_ADDR	CLIENT_EMAIL
11	Hexamedika	14041	Rungkut Surabaya	admin@hexa.co.id
12	Ekalya Vessel	14042	Darmo Surabaya	hrd@ekalya.co.id
13	Avodamitra	14043	Semampir Sidoarjo	hrd@avoda.co.id
14	Golden Union	14044	Jabon Sidoarjo	hr.ga@golden.co.id
15	Orela Shipyard	14045	Pangkah Gresik	support@orela.co.id

Analisa :

Gambar #1

Insert dilakukan pada table **Client** dengan 2 baris pertama menggunakan Single Input menggunakan perintah **INSERT INTO**.

Gambar #2

Insert dilakukan pada table **Client** dengan 3 baris selanjutnya menggunakan Multiple Input menggunakan perintah **INSERT ALL** ditambah pada akhir baris menggunakan klausa **SELECT 1 FROM dual;**

Gambar #3

Hasil dari data yang sudah diinsert atau dimasukkan ke dalam table **Client**

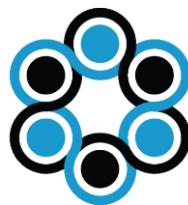
Langkah Ketiga

Insert 5 data pada table **Job** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO job (id_job, id_client, job_name, category, description) values (1101, 11, 'Network Administrator', 'IT', 'Manage and monitoring network');

INSERT INTO job (id_job, id_client, job_name, category, description) values (1102, 12, 'Front-End Developer', 'IT', 'Design and develop UI Website');
```



Laporan Sementara

```
INSERT ALL
INTO job (id_job, id_client, job_name,
category, description) values (1103, 13,
'RAC Engineer', 'Engineer', 'Install and
maintain cooling system')
INTO job (id_job, id_client, job_name,
category, description) values (1104, 14,
'Accounting Staff', 'Finance', 'Reviewing
financial statement')
INTO job (id_job, id_client, job_name,
category, description) values (1105, 15,
'QA/QC Vessel', 'Engineer', 'Inspect and
test materials')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT INTO job (id_job, id_client, job_name, category, description) values (1101, 11, 'Network A
dministrator', 'IT', 'Manage and monitoring network');

1 row created.

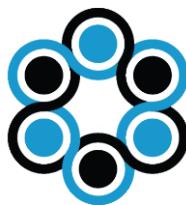
SQL> INSERT INTO job (id_job, id_client, job_name, category, description) values (1102, 12, 'Front-End
Developer', 'IT', 'Design and develop UI Website');

1 row created.

SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1103, 13, 'RAC Engineer',
  3  'Engineer', 'Install and maintain cooling system')
  3  INTO job (id_job, id_client, job_name, category, description) values (1104, 14, 'Accounting Staff',
  4  'Finance', 'Reviewing financial statement')
  4  INTO job (id_job, id_client, job_name, category, description) values (1105, 15, 'QA/QC Vessel',
  5  'Engineer', 'Inspect and test materials')
  5  SELECT 1 FROM dual;

3 rows created.
```

```
SQL> select * from job;
      ID_JOB ID_CLIENT JOB_NAME                                     CATEGORY
      DESCRIPTION
-----+
-----+
      1101      11 Network Administrator                         IT
                  Manage and monitoring network
      1102      12 Front-End Developer                          IT
                  Design and develop UI Website
      1103      13 RAC Engineer                                Engineer
                  Install and maintain cooling system
      1104      14 Accounting Staff                            Finance
                  Reviewing financial statement
      1105      15 QA/QC Vessel                               Engineer
                  Inspect and test materials
```



Laporan Sementara

Analisa :

Gambar #1

Insert dilakukan pada table **Job** dengan 2 baris pertama menggunakan Single Input menggunakan perintah **INSERT INTO**.

Gambar #2

Insert dilakukan pada table **Job** dengan 3 baris selanjutnya menggunakan Multiple Input menggunakan perintah **INSERT ALL** ditambah pada akhir baris menggunakan klausa **SELECT 1 FROM dual**;

Gambar #3

Hasil dari data yang sudah diinsert atau dimasukkan ke dalam table **Job**

Langkah Keempat

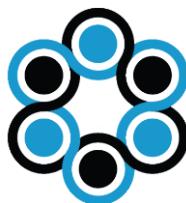
Insert 5 data pada table **Project** (dengan menggunakan Single Insert 2 data) dan (dengan menggunakan Multiple Insert 3 data).

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1101, 101,
to_date('01/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy'));

INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1102, 102,
to_date('02/05/2020','dd/mm/yyyy'),
to_date('10/05/2020', 'dd/mm/yyyy'));

INSERT ALL
INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1103, 103,
to_date('03/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy'))
INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1104, 104,
to_date('04/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy'))
```



Laporan Sementara

```
INSERT ALL
INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1103, 103,
to_date('03/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy'))
INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1104, 104,
to_date('04/05/2020', 'dd/mm/yyyy'),
to_date('20/05/2020', 'dd/mm/yyyy'))
INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1105, 105,
to_date('05/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy'))
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1101, 101, to_date('01/05/2020', 'dd/mm/yyyy'), to_date('20/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1102, 102, to_date('02/05/2020', 'dd/mm/yyyy'), to_date('10/05/2020', 'dd/mm/yyyy'));
1 row created.

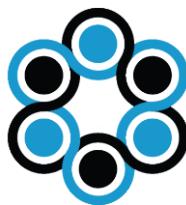
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1103, 103, to_date('03/05/2020', 'dd/mm/yyyy'), to_date('30/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1104, 104, to_date('04/05/2020', 'dd/mm/yyyy'), to_date('20/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1105, 105, to_date('05/05/2020', 'dd/mm/yyyy'), to_date('15/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL>
```

```
SQL> select * from project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----
            3       1101        101 01-MAY-20 20-MAY-20
            4       1102        102 02-MAY-20 10-MAY-20
            6       1103        103 03-MAY-20 30-MAY-20
            7       1104        104 04-MAY-20 20-MAY-20
            8       1105        105 05-MAY-20 15-MAY-20
```



Laporan Sementara

Analisa :

Gambar #1

Insert dilakukan pada table **Project** dengan 2 baris pertama menggunakan Single Input menggunakan perintah **INSERT INTO**.

Gambar #2

Insert dilakukan pada table **Project** dengan 3 baris selanjutnya menggunakan Multiple Input menggunakan perintah **INSERT ALL** ditambah pada akhir baris menggunakan klausula **SELECT 1 FROM dual**;

Gambar #3

Hasil dari data yang sudah diinsert atau dimasukkan ke dalam table **Project**.

Langkah Kelima

Update data berupa **id_client**, **job_name**, dan **description** pada table **Job**

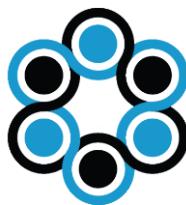
Query

```
UPDATE job
SET id_client = 11,
job_name = 'Network Engineer',
description = 'Maintain and monitoring
network'
WHERE job_name LIKE 'Network%' OR category
= 'IT' AND id_job < 1105;
```

Screenshot

```
SQL> UPDATE job
  2 SET id_client = 11,
  3 job_name = 'Network Engineer',
  4 description = 'Maintain and monitoring network'
  5 WHERE job_name LIKE 'Network%' OR category = 'IT' AND id_job < 1105;
2 rows updated.

SQL> select * from job;
      ID_JOB ID_CLIENT JOB_NAME          DESCRIPTION          CATEGORY
-----  -----  -----  -----
           1101      11 Network Engineer      Maintain and monitoring network      IT
           1102      11 Network Engineer      Maintain and monitoring network      IT
           1103      13 RAC Engineer       Install and maintain cooling system      Engineer
           1104      14 Accounting Staff    Reviewing financial statement      Finance
           1105      15 QA/QC Vessel      Inspect and test materials      Engineer
```



Laporan Sementara

Analisa :

Melakukan update pada table **Job** dengan data yang di update adalah **id_client**, **job_name**, dan **description** dengan kondisi dimana data pada **job_name** memiliki karakteristik kata **Network** di awal kalimat **ATAU** kategori = **IT DAN id_job < 1105**

Langkah Keenam

Update data berupa **employee_name** dan **employee_addr** pada table **Employee**.

Query

```
UPDATE employee
SET employee_name = 'Naurasari',
employee_addr = 'Taman Bungkul Surabaya'
WHERE email LIKE 'naura%' AND employee_name
LIKE 'Naura%' AND id_employee = 103;
```

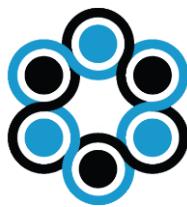
Screenshot

```
SQL> UPDATE employee
  2 SET employee_name = 'Naurasari',
  3 employee_addr = 'Taman Bungkul Surabaya'
  4 WHERE email LIKE 'naura%' AND employee_name LIKE 'Naura%' AND id_employee = 103;
1 row updated.

SQL> select * from employee;
ID_EMPLOYEE EMPLOYEE_NAME          EMPLOYEE_ADDR          GEN PHONE_NO EMAIL
-----  -----
          101 Muchlas           Dusun Duran Sedati      L   8124186 muchlas@g
mail.com
          102 Sterling          Surabaya Sukolilo       L   8225045 sterling@g
mail.com
          103 Naurasari         Taman Bungkul Surabaya P   8774501 naura@gma
il.com
          104 Alya              WR Supratman Surabaya  P   85643801 alya@gmai
l.com
          105 John              Gedangan Sidoarjo     L   811345716 john@gmai
l.com
SQL>
```

Analisa :

(tulis Analisa)



Laporan Sementara

Langkah Ketujuh

Update data berupa **client_name** dan **client_email** pada table **Client**.

Query

```
UPDATE client
SET client_name = 'CV Avodamitra',
client_email = 'hrd@avodamitra.co.id'
WHERE client_name LIKE 'Avoda%' OR
client_email LIKE 'hrd@avo%' AND id_client
< 15;
```

Screenshot

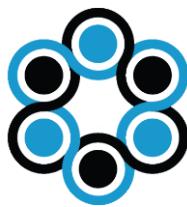
```
SQL> UPDATE client
2 SET client_name = 'CV Avodamitra',
3 client_email = 'hrd@avodamitra.co.id'
4 WHERE client_name LIKE 'Avoda%' OR client_email LIKE 'hrd@avo%' AND id_client < 15;

1 row updated.

SQL> SELECT * FROM client;
ID_CLIENT CLIENT_NAME          CLIENT_ADDR      CLIENT_EMAIL
CLIENT_PHONE_NO
-----
11 Hexamedika           Rungkut Surabaya    admin@hexa.co.id
14041
12 Ekalya Vessel         Darmo Surabaya   hrd@ekalya.co.id
14042
13 CV Avodamitra        Semampir Sidoarjo  hrd@avodamitra.co.id
14 Golden Union          Jabon Sidoarjo   hr.ga@golden.co.id
14043
15 Orela Shipyard        Pangkah Gresik   support@orela.co.id
14044
14045
SQL>
```

Analisa :

(tulis Analisa)



Laporan Sementara

Langkah Kedelapan

Update data berupa **start_date** dan **id_employee** pada table **Project**

Query

```
UPDATE project
SET start_date = to_date('07/05/2020',
'dd/mm/yyyy'),
id_employee = 103
WHERE id_project <= 8 AND start_date =
to_date('05/05/2020', 'dd/mm/yyyy') AND
end_date >= to_date('15/05/2020',
'dd/mm/yyyy');
```

Screenshot

```
SQL> UPDATE project
2 SET start_date = to_date('07/05/2020', 'dd/mm/yyyy'),
3 id_employee = 103
4 WHERE id_project <= 8 AND start_date = to_date('05/05/2020', 'dd/mm/yyyy') AND end_
date >= to_date('15/05/2020', 'dd/mm/yyyy');

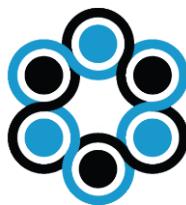
1 row updated.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----  -----
      3          1101      101 01-MAY-20 20-MAY-20
      4          1102      102 02-MAY-20 10-MAY-20
      6          1103      103 03-MAY-20 30-MAY-20
      7          1104      104 04-MAY-20 20-MAY-20
      8          1105      103 07-MAY-20 15-MAY-20

SQL>
```

Analisa :

(tulis Analisa)



Laporan Sementara

Langkah Kesembilan

Delete 3 data pada table **Project** dengan penggunaan kondisi yang berbeda – beda.

Query

```
DELETE FROM project
WHERE start_date = to_date('01/05/2020',
'dd/mm/yyyy') AND end_date =
to_date('20/05/2020', 'dd/mm/yyyy') AND
id_project < 8;

DELETE FROM project
WHERE end_date = to_date('20/05/2020',
'dd/mm/yyyy') AND start_date >
to_date('03/05/2020', 'dd/mm/yyyy') OR
id_project = 7;

DELETE FROM project
WHERE start_date BETWEEN
to_date('02/05/2020', 'dd/mm/yyyy') AND
to_date('06/05/2020', 'dd/mm/yyyy') AND
id_project > 4;
```

Screenshot

```
SQL> DELETE FROM project
  2 WHERE start_date = to_date('01/05/2020', 'dd/mm/yyyy') AND end_date = to_date('20/05/2020', 'dd/mm/yyyy') AND id_project < 8;
1 row deleted.

SQL> DELETE FROM project
  2 WHERE end_date = to_date('20/05/2020', 'dd/mm/yyyy') AND start_date > to_date('03/05/2020', 'dd/mm/yyyy') OR id_project = 7;
1 row deleted.

SQL> DELETE FROM project
  2 WHERE start_date BETWEEN to_date('02/05/2020', 'dd/mm/yyyy') AND to_date('06/05/2020', 'dd/mm/yyyy') AND id_project > 4;
1 row deleted.

SQL>
```



Laporan Sementara

```
SQL> SELECT * FROM project;  
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE  
-----  
        4          1102      102 02-MAY-20 10-MAY-20  
        8          1105      103 07-MAY-20 15-MAY-20  
SQL>
```

Analisa :

(tulis Analisa)

Langkah Kesepuluh

Melakukan perintah SAVEPOINT untuk menyimpan CHECKPOINT dimana data terakhir disimpan.

Query

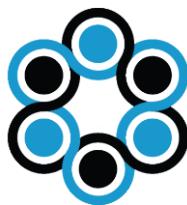
```
SAVEPOINT del_project;
```

Screenshot

```
SQL> SELECT * FROM project;  
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE  
-----  
        4          1102      102 02-MAY-20 10-MAY-20  
        8          1105      103 07-MAY-20 15-MAY-20  
SQL> SAVEPOINT del_project;  
Savepoint created.  
SQL>
```

Analisa :

(tulis Analisa)



Laporan Sementara

Langkah Kesebelas

Melakukan perintah **ROLLBACK** untuk mengembalikan dimana data terakhir disimpan pada pembuatan **SAVEPOINT**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1104, 101,
to_date('10/05/2020', 'dd/mm/yyyy'),
to_date('22/05/2020', 'dd/mm/yyyy'))
ROLLBACK TO SAVEPOINT del_project;
```

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1104, 101, to_date('10/05/2020', 'dd/mm/yyyy'), to_date('22/05/2020',
', 'dd/mm/yyyy'));
1 row created.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----        -----      -----
        4            1102      102 02-MAY-20 10-MAY-20
        8            1105      103 07-MAY-20 15-MAY-20
        9            1104      101 10-MAY-20 22-MAY-20

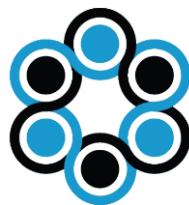
SQL> ROLLBACK TO SAVEPOINT del_project;
Rollback complete.

SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----        -----      -----
        4            1102      102 02-MAY-20 10-MAY-20
        8            1105      103 07-MAY-20 15-MAY-20

SQL>
```

Analisa :

(tulis Analisa)



Laporan Sementara

Langkah Keduabelas

Melakukan perintah **COMMIT** untuk menyimpan secara permanen dari perubahan data yang terakhir.

Query

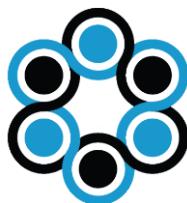
```
COMMIT;
```

Screenshot

```
SQL> SELECT * FROM project;
ID_PROJECT      ID_JOB ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----
        4           1102      102 02-MAY-20 10-MAY-20
        8           1105      103 07-MAY-20 15-MAY-20
SQL> COMMIT;
Commit complete.
SQL>
```

Analisa :

(tulis Analisa)



Laporan Sementara

Langkah Ketigabelas

Melakukan perintah **SELECT** dengan menggunakan atau menerapkan kondisi **ORDER BY** pada table **Project**.

Query

```
SELECT * FROM project ORDER BY end_date;
```

Screenshot

```
SQL> SELECT * FROM project ORDER BY end_date;
ID_PROJECT      ID_JOB  ID_EMPLOYEE START_DATE END_DATE
-----          -----   -----
        4           1102       102 02-MAY-20 10-MAY-20
        8           1105       103 07-MAY-20 15-MAY-20
```

Analisa :

(tulis Analisa)

Langkah Keempatbelas

Melakukan perintah **SELECT** dengan menggunakan atau menerapkan kondisi **GROUP BY** pada table **Project**.

Query

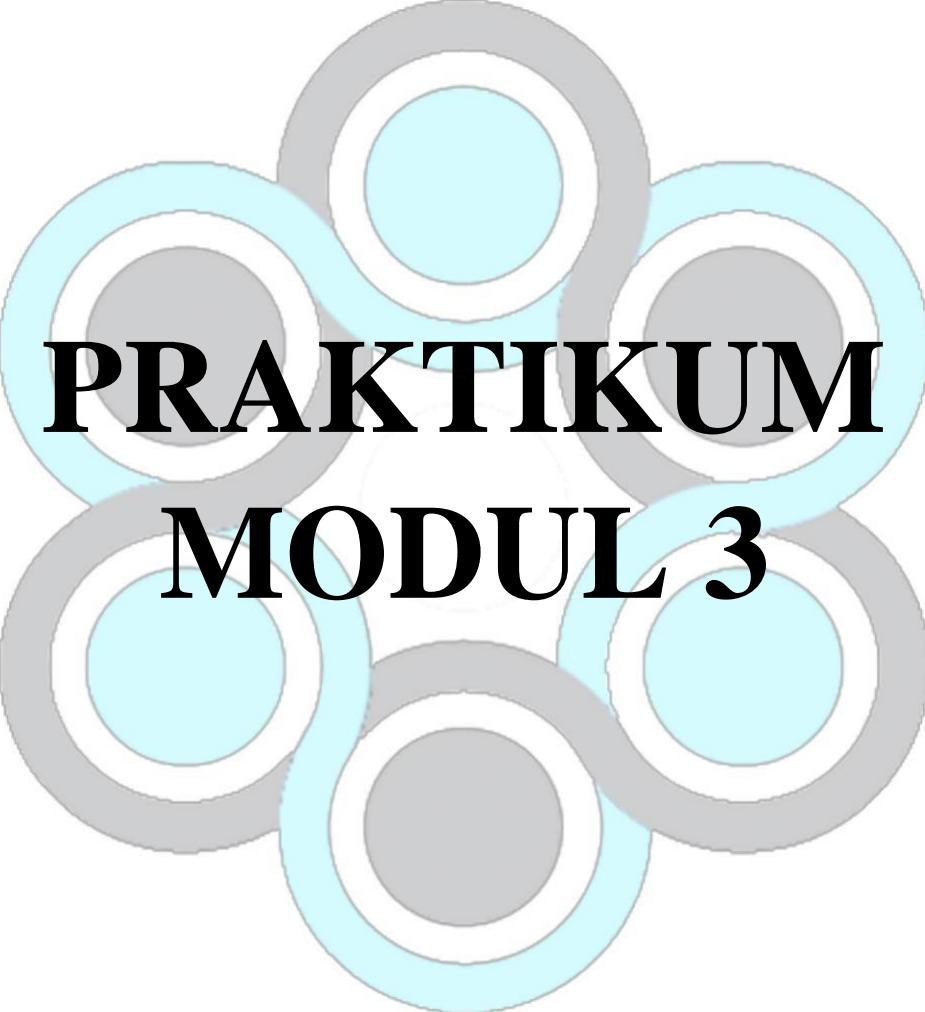
```
SELECT end_date, count(id_project) FROM
project GROUP BY end_date;
```

Screenshot

```
SQL> SELECT end_date, count(id_project) FROM project GROUP BY end_date;
END_DATE  COUNT(ID_PROJECT)
-----
10-MAY-20          1
15-MAY-20          1
SQL>
```

Analisa :

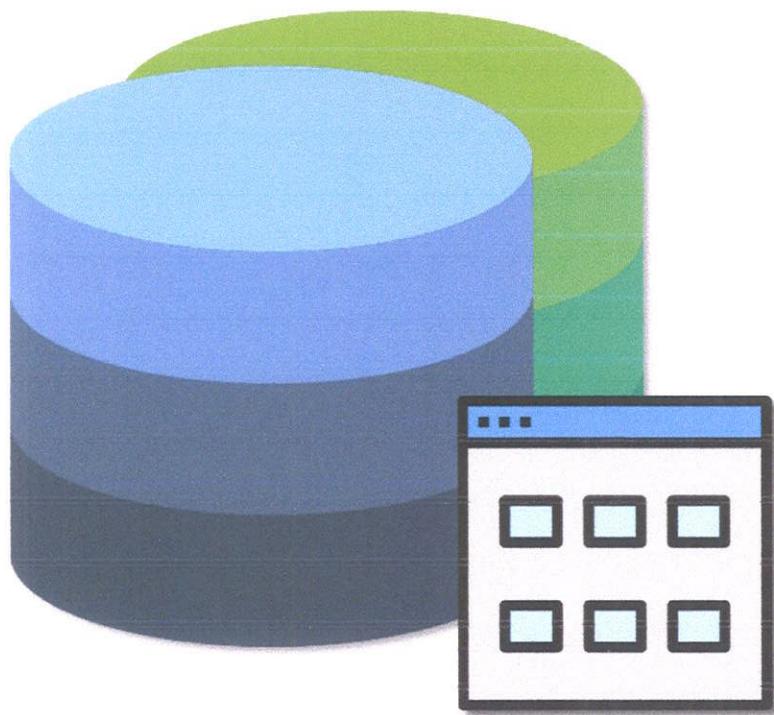
(tulis Analisa)



PRAKTIKUM MODUL 3

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020

LAPORAN PRAKTIKUM

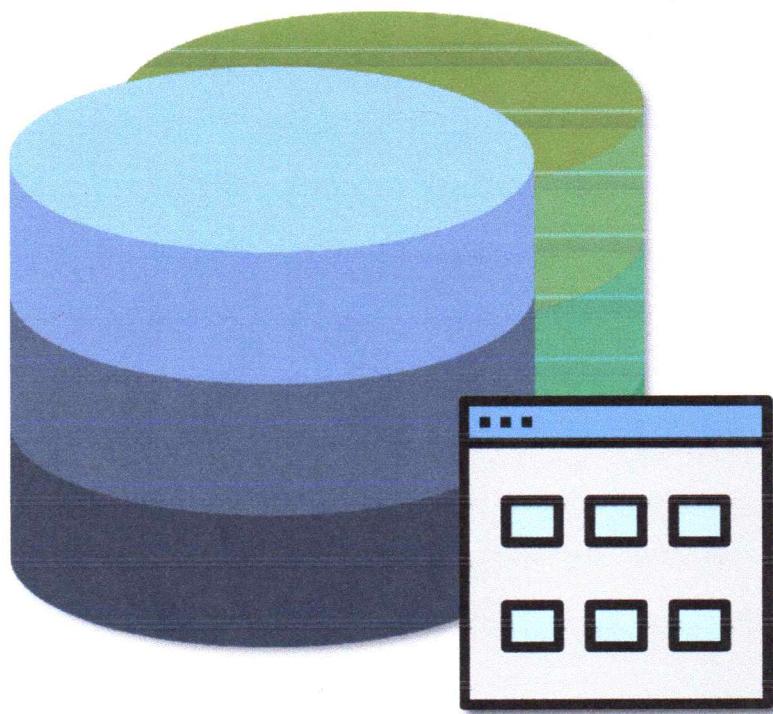


**PRAKTIKUM BASIS DATA
PERIODE V - Tahun 2019/2020**

NAMA : Achmad Muchlasin
NPM : 06.2018.1-06941
MODUL : III

--

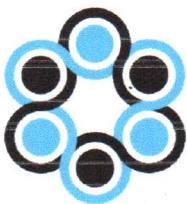
LAPORAN PENDAHULUAN



PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmael Muchlasin
NPM : 06.2018.1.06941
MODUL : II

--



Tugas Pertanyaan

Modul III

Bab IV

3.1 Tujuan

- Praktikan dapat mengimplementasikan konsep agregasi & menggunakan sub query.
- Praktikan dapat menampilkan data dengan sebuah kondisi tertentu dengan menggunakan subquery.

3.2 Kriteria Penilaian

- Praktikan dapat menggunakan fungsi aggregate
- Praktikan dapat mengimplementasikan subquery

3.3 Soal Pendahuluan

- Buatlah sebuah query untuk menampilkan data berikut :

ID_PESERTAIDU TOTAL-ITEM-YANG-DIPESAN

3

3

- Tampilkan data terlebih dahulu pada tabel yang dituju :

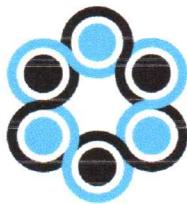
SQL > SELECT * FROM detail_pemesanan_dgut;

ID_PESERTA ID_PEMESANAN JUMLAH_MISKEN

1	1	1	6
---	---	---	---

2	2	2	20
---	---	---	----

3	3	3	30
---	---	---	----



Tugas Pertanyaan

- Tuliskan query seperti dibawah ini

SQL > SELECT id_pemesanan, (SELECT sum(jumlah) FROM detail_pemesanan WHERE id_pemesanan = 06941)

2 WITHETB detail_pemesanan_06941 AS (pemesanan_06941.id_pemesanan =

3 AS total_item_yang_dipesan FROM pemesanan_06941
WHERE id_pemesanan = 3;

2. Buatlah sebuah query untuk menampilkan data berikut :

id_kategori	TERBARU	TERENAH
-------------	---------	---------

1	90	10
---	----	----

2	40	40
---	----	----

3	35	35
---	----	----

4	95	95
---	----	----

5	100	100
---	-----	-----

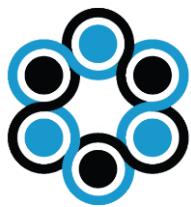
- Query untuk menampilkan seperti diatas adalah :

SQL > SELECT id_kategori, MAX(stk_produk) AS

2 terbaru, MIN(stk_produk) AS terendah FROM produk - c6941

3 ORDER BY id_kategori;

- Pada query dratas akan merekap dari menampilkan nilai terendah dan tertinggi stok produk yang tersedia di tabel produk.



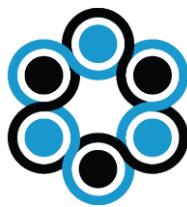
Tugas Praktikum

Soal Praktikum

1. Insert minimal 5 data pada setiap tabel

2. Buatlah query dengan menerapkan fungsi aggregate :
 - a. Max dan Min dalam 1 Query
 - b. Count, 1 klausa dan 2 operator dalam 1 Query
 - c. Sum dalam 1 Query
 - d. Average pada soal 2C diatas

3. Buatlah Subquery dengan menerapkan :
 - a. Max dan Min
 - b. Count, 1 klausa dan 2 operator
 - c. Sum
 - d. Average
 - e. Group by
 - f. Select setelah where
 - g. Minimal 3 nested query



Tugas Praktikum

Langkah Pertama

Insert minimal 5 data pada table **Employee** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (106, 'Maya', 'Medaeng', 'P',
'1066666', 'maya@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (107, 'Aga', 'Waru', 'L', '1077777',
'aga@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (108, 'Nara', 'Sidoarjo', 'P',
'1088888', 'nara@gmail.com')
```

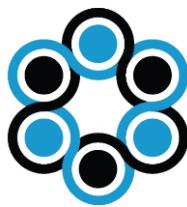
```
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (109, 'Moka', 'Candi', 'L',
'1099999', 'moka@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (110, 'Tara', 'Tanggulangin', 'L',
'1100000', 'tara@gmail.com')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(106, 'Maya', 'Medaeng', 'P', '1066666', 'maya@gmail.com')
  3  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(107, 'Aga', 'Waru', 'L', '1077777', 'aga@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(108, 'Nara', 'Sidoarjo', 'P', '1088888', 'nara@gmail.com')
  5  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(109, 'Moka', 'Candi', 'L', '1099999', 'moka@gmail.com')
  6  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(110, 'Tara', 'Tanggulangin', 'L', '1100000', 'tara@gmail.com')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Tugas Praktikum

Analisa

Insert dilakukan pada table **Employee** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Kedua

Insert minimal 5 data pada table **Client** menggunakan **INSERT ALL**.

Query

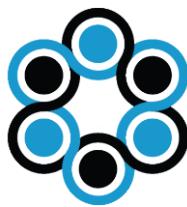
```
INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (16, 'Foxtrot Alto', 'Bukti Darmo',
'hr.ga@foxalto.co', '14444')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (17, 'Padinet', 'Mayjen Sungkono',
'admin.ho@padi.net', '15555')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (18, 'Grand Tower Corp', 'Basuki
Rahmat', 'admin.hr@gtower.co.id', '16666')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (19, 'PT Sejahtera Abadi',
'Tanggulangin Sda', 'inquiry@sa.co.id',
'17777')
```

Screenshot

```
SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  6, 'Foxtrot Alto', 'Bukti Darmo', 'hr.ga@foxalto.co', '14444')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  7, 'Padinet', 'Mayjen Sungkono', 'admin.ho@padi.net', '15555')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  8, 'Grand Tower Corp', 'Basuki Rahmat', 'admin.hr@gtower.co.id', '16666')
  5  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  9, 'PT Sejahtera Abadi', 'Tanggulangin Sda', 'inquiry@sa.co.id', '17777')
  6  INTO client (id_client, client_name, client_addr, client_email, client_email, client_phone_no) values (2
  0, 'Maersk Corp', 'Pandaan', 'hr.ho@maersk.co.id', '18888')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Tugas Praktikum

Analisa :

Insert dilakukan pada table **Client** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Ketiga

Insert minimal 5 data pada table **Job** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (16, 'Foxtrot Alto', 'Bukti Darmo',
    'hr.ga@foxalto.co', '14444')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (17, 'Padinet', 'Mayjen Sungkono',
    'admin.ho@padi.net', '15555')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (18, 'Grand Tower Corp', 'Basuki
    Rahmat', 'admin.hr@gtower.co.id', '16666')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (19, 'PT Sejahtera Abadi',
    'Tanggulangin Sda', 'inquiry@sa.co.id',
    '17777')
```

Screenshot

```
SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1106, 16, 'Software
Engineer', 'IT engineer', 'Maintain website')
  3  INTO job (id_job, id_client, job_name, category, description) values (1107, 18, 'Bussiness
Dev', 'General', 'Collect and submit tender')
  4  INTO job (id_job, id_client, job_name, category, description) values (1108, 16, 'IT Suppor
t', 'IT engineer', 'Maintain network peripheral')
  5  INTO job (id_job, id_client, job_name, category, description) values (1109, 20, 'HRD', 'HR
', 'Manage employee')
  6  INTO job (id_job, id_client, job_name, category, description) values (1110, 20, 'General A
ffair', 'General', 'Manage office tools')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Tugas Praktikum

Analisa :

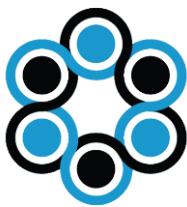
Insert dilakukan pada table **Job** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Keempat

Insert minimal 5 data pada table **Project** menggunakan **INSERT INTO**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1106, 106,
to_date('06/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1107, 107,
to_date('07/05/2020', 'dd/mm/yyyy'),
to_date('18/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1108, 108,
to_date('08/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1109, 107,
to_date('09/05/2020', 'dd/mm/yyyy'),
to_date('21/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1110, 110,
to_date('10/05/2020', 'dd/mm/yyyy'),
to_date('24/05/2020', 'dd/mm/yyyy'));
```



Tugas Praktikum

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1106, 106, to_date('06/05/2020', 'dd/mm/yyyy'), to_date('15/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1107, 107, to_date('07/05/2020', 'dd/mm/yyyy'), to_date('18/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1108, 108, to_date('08/05/2020', 'dd/mm/yyyy'), to_date('30/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1109, 107, to_date('09/05/2020', 'dd/mm/yyyy'), to_date('21/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1110, 110, to_date('10/05/2020', 'dd/mm/yyyy'), to_date('14/05/2020', 'dd/mm/yyyy'));  
1 row created.  
SQL>
```

Analisa :

Insert dilakukan pada table **Project** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Kelima

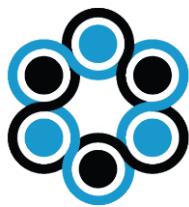
Membuat query dengan menerapkan fungsi aggregate **MIN** dan **MAX** dalam 1 query.

Query

```
SELECT MIN(id_project) AS  
      id_project_terendah, MAX(id_project) as  
      id_project_tertinggi FROM project;
```

Screenshot

```
SQL> select min(id_project) as id_project_terendah,  
 2   max(id_project) as id_project_tertinggi from project;  
  
ID_PROJECT_TERENDAH ID_PROJECT_TERTINGGI  
-----  
2                   25  
  
SQL>
```



Tugas Praktikum

Analisa :

Menggunakan fungsi **MIN** untuk menampilkan **id_project_terendah** dan fungsi **MAX** untuk menampilkan **id_project_tertinggi**.

Langkah Keenam

Membuat query dengan menerapkan fungsi aggregate **Count**, 1 klausa dan 2 operator dalam 1 Query.

Query

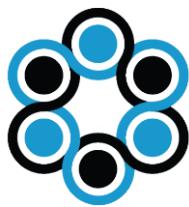
```
select count(id_project) as  
project_dibawah_tgl_8_mei from project  
where start_date < to_date('08/05/2020',  
'dd/mm/yyyy') AND id_project > 2;
```

Screenshot

```
SQL> select count(id_project) as project_dibawah_tgl_8_mei from project  
2 where start_date < to_date('08/05/2020', 'dd/mm/yyyy')  
3 AND id_project > 2;  
  
PROJECT_DIBAWAH_TGL_8_MEI  
-----  
3  
  
SQL>
```

Analisa :

Menggunakan fungsi **COUNT** untuk menghitung jumlah project yang ada pada table **Project** dengan mengambil atau menghitung data berdasarkan **id_project**. Dimana data yang dicari memiliki kondisi tanggal di bawah **8 Mei 2020** dan **id_project** lebih dari 2.



Tugas Praktikum

Langkah Ketujuh

Membuat query dengan menerapkan fungsi aggregate **SUM** dalam 1 Query.

Query

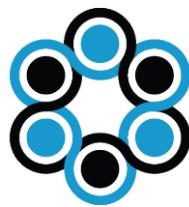
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT SUM(id_client) AS  
2 total_id_client FROM job;  
  
TOTAL_ID_CLIENT  
-----  
157  
  
SQL>
```

Analisa :

Menggunakan fungsi **SUM** untuk menghitung akumulasi dari **id_client** sehingga diperoleh **total_id_client** pada table **Job** dalam 1 line query.



Tugas Praktikum

Langkah Kedelapan

Membuat query dengan menerapkan fungsi aggregate **AVERAGE** pada soal 2C diatas

Query

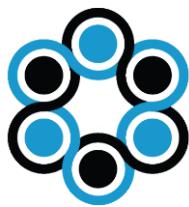
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT AVG(id_client) AS  
2 rerata_total_id_client FROM job;  
  
RERATA_TOTAL_ID_CLIENT  
-----  
15.7  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menghitung rata – rata dari soal nomer 2C yakni jumlah dari **id_client** dan menghasilkan rata – rata dari jumlah tersebut pada table **Job**.



Tugas Praktikum

Langkah Kesembilan

Membuat query dengan menerapkan fungsi aggregate **AVERAGE** pada soal 2C diatas

Query

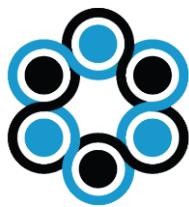
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT AVG(id_client) AS  
2  rerata_total_id_client FROM job;  
  
RERATA_TOTAL_ID_CLIENT  
-----  
          15.7  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menghitung rata – rata dari soal nomer 2C yakni jumlah dari **id_client** dan menghasilkan rata – rata dari jumlah tersebut pada table **Job**.



Tugas Praktikum

Langkah Kesepuluh

Membuat subquery dengan menerapkan fungsi aggregate **MAX** dan **MIN**.

Query

```
SELECT id_job, (SELECT MAX(id_client) FROM
job)
AS id_client_max, (SELECT MIN(id_client) FROM
job
AS id_client_min FROM job
WHERE id_client = 20 OR id_client = 11;
```

Screenshot

```
SQL> SELECT id_job, (SELECT MAX(id_client) FROM job)
  2 AS id_client_max, (SELECT MIN(id_client) FROM job)
  3 AS id_client_min FROM job
  4 WHERE id_client = 20 OR id_client = 11;

ID_JOB ID_CLIENT_MAX ID_CLIENT_MIN
----- ----- -----
 1109      20          11
 1110      20          11
 1101      20          11
```

Analisa :

Menggunakan fungsi **MAX** dan **MIN** dengan penambahan penggunaan **Subquery** untuk mencari **id_client_max** dan **id_client_min** pada table **Job**. Dimana data yang ditampilkan memiliki kondisi **id_client = 20 ATAU id_client = 11**.



Tugas Praktikum

Langkah Kesebelas

Membuat subquery dengan menerapkan fungsi aggregate **Count**, 1 klausula dan 2 operator.

Query

```
SELECT id_client, (SELECT COUNT(id_client)
FROM job
WHERE job.id_client = client.id_client)
AS total_id_client FROM client WHERE
id_client > 10 AND client_name LIKE 'F%' OR
client_name LIKE 'P%';
```

Screenshot

```
SQL> SELECT id_client, (SELECT COUNT(id_client) FROM job
  2 WHERE job.id_client = client.id_client)
  3 AS jumlah_client from client WHERE id_client > 10 AND client_name LIKE 'F%'
  4 OR client_name LIKE 'P%';

ID_CLIENT JUMLAH_CLIENT
-----
16          2
17          0
19          0

SQL>
```

Analisa :

Menggunakan fungsi **COUNT** dengan penambahan penggunaan **Subquery** untuk menampilkan jumlah client pada table **Client**. Dimana data yang ditampilkan memiliki kondisi **id_client** lebih dari **10 DAN client_name** memiliki karakteristik berhuruf depan **F ATAU P**.



Tugas Praktikum

Langkah Kedua

Membuat subquery dengan menerapkan fungsi aggregate **SUM**.

Query

```
SELECT id_job, (SELECT SUM(id_job) FROM project  
WHERE project.id_job = job.id_job)  
AS jumlah_id_job FROM job;
```

Screenshot

```
SQL> SELECT id_job, (SELECT SUM(id_job) FROM project  
2 WHERE project.id_job = job.id_job)  
3 AS jumlah_id_job FROM job;  
  
ID_JOB JUMLAH_ID_JOB  
-----  
1101      1101  
1102      1102  
1103  
1104  
1105  
1106      1106  
1107      1107  
1108      1108  
1109      1109  
1110      1110  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **SUM** untuk menghitung akumulasi dari **id_job** pada table **Job** dengan menggunakan alias **jumlah_id_job**.



Tugas Praktikum

Langkah Ketigabelas

Membuat subquery dengan menerapkan fungsi aggregate **AVERAGE**

Query

```
SELECT id_client, (SELECT AVG(id_client) FROM job  
WHERE job.id_client = client.id_client)  
AS rerata_id_client FROM client;
```

Screenshot

```
SQL> SELECT id_client, (SELECT AVG(id_client) FROM job  
2 WHERE job.id_client = client.id_client)  
3 AS rerata_id_client FROM client;  
  
ID_CLIENT RERATA_ID_CLIENT  
-----  
11          11  
12          12  
13  
14          14  
15          15  
16          16  
17  
18          18  
19  
20          20  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menampilkan data berupa rata – rata dari akumulasi **id_client** pada table **Client** dengan menggunakan alias **rerata_id_client**.



Tugas Praktikum

Langkah Keempatbelas

Membuat subquery dengan menerapkan fungsi **GROUP BY**.

Query

```
SELECT id_employee, (SELECT  
COUNT(id_employee) FROM project  
WHERE project.id_employee =  
employee.id_employee)  
AS jumlah_id_employee FROM employee  
GROUP BY id_employee;
```

Screenshot

```
SQL> SELECT id_employee, (SELECT COUNT(id_employee) FROM project  
2 WHERE project.id_employee = employee.id_employee)  
3 AS jumlah_id_employee FROM employee  
4 GROUP BY id_employee;  
  
ID_EMPLOYEE JUMLAH_ID_EMPLOYEE  
-----  
101          0  
102          2  
103          0  
104          0  
105          0  
106          1  
107          2  
108          1  
109          0  
110          1  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **GROUP BY** untuk menampilkan **id_employee** dan **jumlah_id_employee** pada table **Project**. Dimana data yang ditampilkan memiliki kondisi



Tugas Praktikum

Langkah Kelimabelas

Membuat subquery dengan menerapkan fungsi **SELECT** setelah klausu **WHERE**.

Query

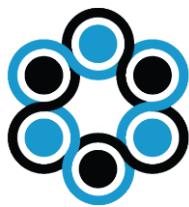
```
SELECT id_project, MAX(start_date) AS  
tertinggi, MIN(start_date) AS terendah FROM  
project  
WHERE id_project IN (SELECT id_project FROM  
project WHERE start_date >  
to_date('06/05/2020', 'dd/mm/yyyy'))  
GROUP BY id_project;
```

Screenshot

```
SQL> SELECT id_project, MAX(start_date) AS  
2  tertinggi, MIN(start_date) AS terendah FROM project  
3  WHERE id_project IN (SELECT id_project FROM project WHERE start_date > to_date('06/05/2020  
4  GROUP BY id_project;  
  
ID_PROJECT TERTINGGI TERENDAH  
-----  
      5 07-MAY-20 07-MAY-20  
     22 07-MAY-20 07-MAY-20  
    23 08-MAY-20 08-MAY-20  
    24 09-MAY-20 09-MAY-20  
    25 10-MAY-20 10-MAY-20  
  
SQL>
```

Analisa :

Menggunakan fungsi **SELECT** setelah klausu **WHERE** pada **Subquery** untuk menampilkan **id_project**, **tanggal_tertinggi**, **tanggal_terendah**. Dimana data yang ditampilkan memiliki kondisi **start_date** lebih dari **6 Mei 2020** ditambah dengan penggunaan **GROUP BY** untuk mengelompokkan data berdasarkan **id_project**.



Tugas Praktikum

Langkah Keenambelas

Membuat subquery dengan menggunakan minimal 3 nested query.

Query

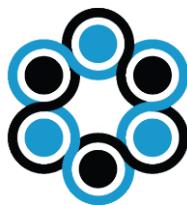
```
SELECT id_project, (SELECT SUM(id_job) FROM project WHERE id_project < (SELECT COUNT(id_project) AS jumlah_id_project FROM project)) AS akumulasi_id_job FROM project GROUP BY id_project;
```

Screenshot

```
SQL> SELECT id_project, (SELECT SUM(id_job) FROM project WHERE id_project < (SELECT COUNT(id_project) AS jumlah_id_project FROM project)) AS akumulasi_id_job FROM project GROUP BY id_project;
ID_PROJECT AKUMULASI_ID_JOB
-----
      2          2203
      5          2203
     21          2203
     22          2203
     23          2203
     24          2203
     25          2203
7 rows selected.
SQL>
```

Analisa :

Menggunakan klausua **SELECT** dengan 3 nested query pada pembuatan **Subquery** yang digunakan untuk menampilkan data **id_project** dan **akumulasi_id_job** pada table **Project**. Dimana data yang ditampilkan memiliki kondisi dimana **id_project** harus kurang dari **jumlah_id_project**. Lalu disertai penggunaan **GROUP BY** untuk mengelompokkan data berdasarkan **id_project**.



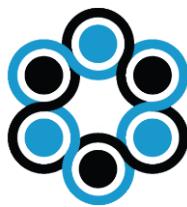
Laporan Sementara

Soal Praktikum

1. Insert minimal 5 data pada setiap tabel

2. Buatlah query dengan menerapkan fungsi aggregate :
 - a. Max dan Min dalam 1 Query
 - b. Count, 1 klausa dan 2 operator dalam 1 Query
 - c. Sum dalam 1 Query
 - d. Average pada soal 2C diatas

3. Buatlah Subquery dengan menerapkan :
 - a. Max dan Min
 - b. Count, 1 klausa dan 2 operator
 - c. Sum
 - d. Average
 - e. Group by
 - f. Select setelah where
 - g. Minimal 3 nested query



Laporan Sementara

Langkah Pertama

Insert minimal 5 data pada table **Employee** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (106, 'Maya', 'Medaeng', 'P',
'1066666', 'maya@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (107, 'Aga', 'Waru', 'L', '1077777',
'aga@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (108, 'Nara', 'Sidoarjo', 'P',
'1088888', 'nara@gmail.com')
```

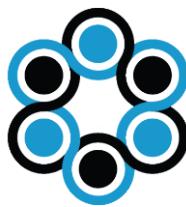
```
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (109, 'Moka', 'Candi', 'L',
'1099999', 'moka@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (110, 'Tara', 'Tanggulangin', 'L',
'1100000', 'tara@gmail.com')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(106, 'Maya', 'Medaeng', 'P', '1066666', 'maya@gmail.com')
  3  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(107, 'Aga', 'Waru', 'L', '1077777', 'aga@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(108, 'Nara', 'Sidoarjo', 'P', '1088888', 'nara@gmail.com')
  5  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(109, 'Moka', 'Candi', 'L', '1099999', 'moka@gmail.com')
  6  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email) values
(110, 'Tara', 'Tanggulangin', 'L', '1100000', 'tara@gmail.com')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Laporan Sementara

Analisa

Insert dilakukan pada table **Employee** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Kedua

Insert minimal 5 data pada table **Client** menggunakan **INSERT ALL**.

Query

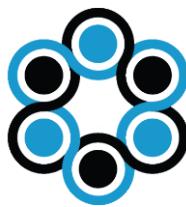
```
INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (16, 'Foxtrot Alto', 'Bukti Darmo',
'hr.ga@foxalto.co', '14444')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (17, 'Padinet', 'Mayjen Sungkono',
'admin.ho@padi.net', '15555')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (18, 'Grand Tower Corp', 'Basuki
Rahmat', 'admin.hr@gtower.co.id', '16666')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (19, 'PT Sejahtera Abadi',
'Tanggulangin Sda', 'inquiry@sa.co.id',
'17777')
```

Screenshot

```
SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  6, 'Foxtrot Alto', 'Bukti Darmo', 'hr.ga@foxalto.co', '14444')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  7, 'Padinet', 'Mayjen Sungkono', 'admin.ho@padi.net', '15555')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  8, 'Grand Tower Corp', 'Basuki Rahmat', 'admin.hr@gtower.co.id', '16666')
  5  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  9, 'PT Sejahtera Abadi', 'Tanggulangin Sda', 'inquiry@sa.co.id', '17777')
  6  INTO client (id_client, client_name, client_addr, client_email, client_email, client_phone_no) values (2
  0, 'Maersk Corp', 'Pandaan', 'hr.ho@maersk.co.id', '18888')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Laporan Sementara

Analisa :

Insert dilakukan pada table **Client** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Ketiga

Insert minimal 5 data pada table **Job** menggunakan **INSERT ALL**.

Query

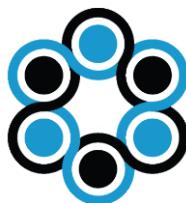
```
INSERT ALL
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (16, 'Foxtrot Alto', 'Bukti Darmo',
    'hr.ga@foxalto.co', '14444')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (17, 'Padinet', 'Mayjen Sungkono',
    'admin.ho@padi.net', '15555')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (18, 'Grand Tower Corp', 'Basuki
    Rahmat', 'admin.hr@gtower.co.id', '16666')
  INTO client (id_client, client_name,
    client_addr, client_email, client_phone_no)
  values (19, 'PT Sejahtera Abadi',
    'Tanggulangin Sda', 'inquiry@sa.co.id',
    '17777')
```

Screenshot

```
SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1106, 16, 'Software
Engineer', 'IT engineer', 'Maintain website')
  3  INTO job (id_job, id_client, job_name, category, description) values (1107, 18, 'Bussiness
Dev', 'General', 'Collect and submit tender')
  4  INTO job (id_job, id_client, job_name, category, description) values (1108, 16, 'IT Suppor
t', 'IT engineer', 'Maintain network peripheral')
  5  INTO job (id_job, id_client, job_name, category, description) values (1109, 20, 'HRD', 'HR
', 'Manage employee')
  6  INTO job (id_job, id_client, job_name, category, description) values (1110, 20, 'General A
ffair', 'General', 'Manage office tools')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```



Laporan Sementara

Analisa :

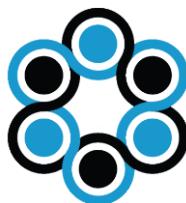
Insert dilakukan pada table **Job** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Keempat

Insert minimal 5 data pada table **Project** menggunakan **INSERT INTO**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1106, 106,
to_date('06/05/2020', 'dd/mm/yyyy'),
to_date('15/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1107, 107,
to_date('07/05/2020', 'dd/mm/yyyy'),
to_date('18/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1108, 108,
to_date('08/05/2020', 'dd/mm/yyyy'),
to_date('30/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1109, 107,
to_date('09/05/2020', 'dd/mm/yyyy'),
to_date('21/05/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1110, 110,
to_date('10/05/2020', 'dd/mm/yyyy'),
to_date('24/05/2020', 'dd/mm/yyyy'));
```



Laporan Sementara

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1106, 106, to_date('06/05/2020', 'dd/mm/yyyy'), to_date('15/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1107, 107, to_date('07/05/2020', 'dd/mm/yyyy'), to_date('18/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1108, 108, to_date('08/05/2020', 'dd/mm/yyyy'), to_date('30/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1109, 107, to_date('09/05/2020', 'dd/mm/yyyy'), to_date('21/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values (id_project.nextval, 1110, 110, to_date('10/05/2020', 'dd/mm/yyyy'), to_date('14/05/2020', 'dd/mm/yyyy'));
1 row created.

SQL>
```

Analisa :

Insert dilakukan pada table **Project** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.

Langkah Kelima

Membuat query dengan menerapkan fungsi aggregate **MIN** dan **MAX** dalam 1 query.

Query

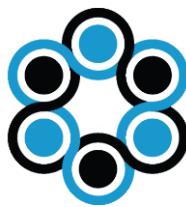
```
SELECT MIN(id_project) AS
       id_project_terendah, MAX(id_project) as
       id_project_tertinggi FROM project;
```

Screenshot

```
SQL> select min(id_project) as id_project_terendah,
      2 max(id_project) as id_project_tertinggi from project;

ID_PROJECT_TERENDAH ID_PROJECT_TERTINGGI
----- ----- 
          2             25

SQL>
```



Laporan Sementara

Analisa :

Menggunakan fungsi **MIN** untuk menampilkan **id_project_terendah** dan fungsi **MAX** untuk menampilkan **id_project_tertinggi**.

Langkah Keenam

Membuat query dengan menerapkan fungsi aggregate **Count**, 1 klausa dan 2 operator dalam 1 Query.

Query

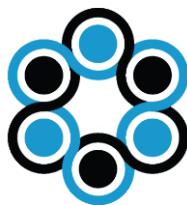
```
select count(id_project) as  
project_dibawah_tgl_8_mei from project  
where start_date < to_date('08/05/2020',  
'dd/mm/yyyy') AND id_project > 2;
```

Screenshot

```
SQL> select count(id_project) as project_dibawah_tgl_8_mei from project  
2 where start_date < to_date('08/05/2020', 'dd/mm/yyyy')  
3 AND id_project > 2;  
  
PROJECT_DIBAWAH_TGL_8_MEI  
-----  
3  
  
SQL>
```

Analisa :

Menggunakan fungsi **COUNT** untuk menghitung jumlah project yang ada pada table **Project** dengan mengambil atau menghitung data berdasarkan **id_project**. Dimana data yang dicari memiliki kondisi tanggal di bawah **8 Mei 2020** dan **id_project** lebih dari 2.



Laporan Sementara

Langkah Ketujuh

Membuat query dengan menerapkan fungsi aggregate **SUM** dalam 1 Query.

Query

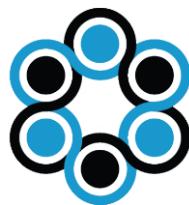
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT SUM(id_client) AS  
2 total_id_client FROM job;  
  
TOTAL_ID_CLIENT  
-----  
          157  
  
SQL>
```

Analisa :

Menggunakan fungsi **SUM** untuk menghitung akumulasi dari **id_client** sehingga diperoleh **total_id_client** pada table **Job** dalam 1 line query.



Laporan Sementara

Langkah Kedelapan

Membuat query dengan menerapkan fungsi aggregate **AVERAGE** pada soal 2C diatas

Query

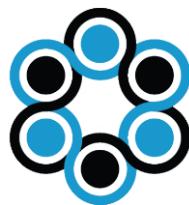
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT AVG(id_client) AS  
2  rerata_total_id_client FROM job;  
  
RERATA_TOTAL_ID_CLIENT  
-----  
15.7  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menghitung rata – rata dari soal nomer 2C yakni jumlah dari **id_client** dan menghasilkan rata – rata dari jumlah tersebut pada table **Job**.



Laporan Sementara

Langkah Kesembilan

Membuat query dengan menerapkan fungsi aggregate **AVERAGE** pada soal 2C diatas

Query

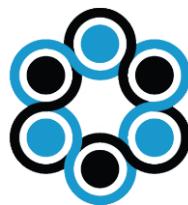
```
SELECT SUM(id_client) AS  
total_id_client FROM job;
```

Screenshot

```
SQL> SELECT AVG(id_client) AS  
2  rerata_total_id_client FROM job;  
  
RERATA_TOTAL_ID_CLIENT  
-----  
          15.7  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menghitung rata – rata dari soal nomer 2C yakni jumlah dari **id_client** dan menghasilkan rata – rata dari jumlah tersebut pada table **Job**.



Laporan Sementara

Langkah Kesepuluh

Membuat subquery dengan menerapkan fungsi aggregate **MAX** dan **MIN**.

Query

```
SELECT id_job, (SELECT MAX(id_client) FROM
job)
AS id_client_max, (SELECT MIN(id_client) FROM
job
AS id_client_min FROM job
WHERE id_client = 20 OR id_client = 11;
```

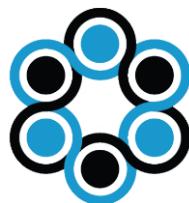
Screenshot

```
SQL> SELECT id_job, (SELECT MAX(id_client) FROM job)
  2 AS id_client_max, (SELECT MIN(id_client) FROM job)
  3 AS id_client_min FROM job
  4 WHERE id_client = 20 OR id_client = 11;

ID_JOB ID_CLIENT_MAX ID_CLIENT_MIN
----- ----- -----
 1109      20          11
 1110      20          11
 1101      20          11
```

Analisa :

Menggunakan fungsi **MAX** dan **MIN** dengan penambahan penggunaan **Subquery** untuk mencari **id_client_max** dan **id_client_min** pada table **Job**. Dimana data yang ditampilkan memiliki kondisi **id_client = 20 ATAU id_client = 11**.



Laporan Sementara

Langkah Kesebelas

Membuat subquery dengan menerapkan fungsi aggregate **Count**, 1 klausula dan 2 operator.

Query

```
SELECT id_client, (SELECT COUNT(id_client)
FROM job
WHERE job.id_client = client.id_client)
AS total_id_client FROM client WHERE
id_client > 10 AND client_name LIKE 'F%' OR
client_name LIKE 'P%';
```

Screenshot

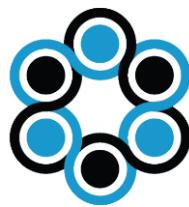
```
SQL> SELECT id_client, (SELECT COUNT(id_client) FROM job
  2 WHERE job.id_client = client.id_client)
  3 AS jumlah_client from client WHERE id_client > 10 AND client_name LIKE 'F%'
  4 OR client_name LIKE 'P%';

ID_CLIENT JUMLAH_CLIENT
-----
16          2
17          0
19          0

SQL>
```

Analisa :

Menggunakan fungsi **COUNT** dengan penambahan penggunaan **Subquery** untuk menampilkan jumlah client pada table **Client**. Dimana data yang ditampilkan memiliki kondisi **id_client** lebih dari **10 DAN client_name** memiliki karakteristik berhuruf depan **F ATAU P**.



Laporan Sementara

Langkah Kedua

Membuat subquery dengan menerapkan fungsi aggregate **SUM**.

Query

```
SELECT id_job, (SELECT SUM(id_job) FROM project  
WHERE project.id_job = job.id_job)  
AS jumlah_id_job FROM job;
```

Screenshot

```
SQL> SELECT id_job, (SELECT SUM(id_job) FROM project  
2 WHERE project.id_job = job.id_job)  
3 AS jumlah_id_job FROM job;  
  
ID_JOB JUMLAH_ID_JOB  
-----  
1101      1101  
1102      1102  
1103  
1104  
1105  
1106      1106  
1107      1107  
1108      1108  
1109      1109  
1110      1110  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **SUM** untuk menghitung akumulasi dari **id_job** pada table **Job** dengan menggunakan alias **jumlah_id_job**.



Laporan Sementara

Langkah Ketigabelas

Membuat subquery dengan menerapkan fungsi aggregate **AVERAGE**

Query

```
SELECT id_client, (SELECT AVG(id_client) FROM job  
WHERE job.id_client = client.id_client)  
AS rerata_id_client FROM client;
```

Screenshot

```
SQL> SELECT id_client, (SELECT AVG(id_client) FROM job  
2 WHERE job.id_client = client.id_client)  
3 AS rerata_id_client FROM client;  
  
ID_CLIENT RERATA_ID_CLIENT  
-----  
11          11  
12          12  
13  
14          14  
15          15  
16          16  
17  
18          18  
19  
20          20  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **AVERAGE** untuk menampilkan data berupa rata – rata dari akumulasi **id_client** pada table **Client** dengan menggunakan alias **rerata_id_client**.



Laporan Sementara

Langkah Keempatbelas

Membuat subquery dengan menerapkan fungsi **GROUP BY**.

Query

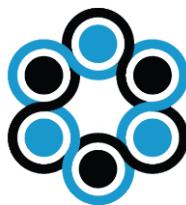
```
SELECT id_employee, (SELECT  
COUNT(id_employee) FROM project  
WHERE project.id_employee =  
employee.id_employee)  
AS jumlah_id_employee FROM employee  
GROUP BY id_employee;
```

Screenshot

```
SQL> SELECT id_employee, (SELECT COUNT(id_employee) FROM project  
2 WHERE project.id_employee = employee.id_employee)  
3 AS jumlah_id_employee FROM employee  
4 GROUP BY id_employee;  
  
ID_EMPLOYEE JUMLAH_ID_EMPLOYEE  
-----  
101          0  
102          2  
103          0  
104          0  
105          0  
106          1  
107          2  
108          1  
109          0  
110          1  
  
10 rows selected.  
  
SQL>
```

Analisa :

Menggunakan fungsi **GROUP BY** untuk menampilkan **id_employee** dan **jumlah_id_employee** pada table **Project**. Dimana data yang ditampilkan memiliki kondisi



Laporan Sementara

Langkah Kelimabelas

Membuat subquery dengan menerapkan fungsi **SELECT** setelah klausa **WHERE**.

Query

```
SELECT id_project, MAX(start_date) AS tertinggi, MIN(start_date) AS terendah FROM project
WHERE id_project IN (SELECT id_project FROM project WHERE start_date >
to_date('06/05/2020', 'dd/mm/yyyy'))
GROUP BY id_project;
```

Screenshot

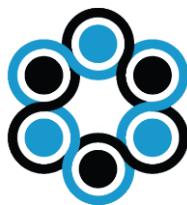
```
SQL> SELECT id_project, MAX(start_date) AS
2 tertinggi, MIN(start_date) AS terendah FROM project
3 WHERE id_project IN (SELECT id_project FROM project WHERE start_date > to_date('06/05/2020
', 'dd/mm/yyyy'))
4 GROUP BY id_project;

ID_PROJECT TERTINGGI TERENDAH
-----
      5 07-MAY-20 07-MAY-20
     22 07-MAY-20 07-MAY-20
     23 08-MAY-20 08-MAY-20
     24 09-MAY-20 09-MAY-20
     25 10-MAY-20 10-MAY-20

SQL>
```

Analisa :

Menggunakan fungsi **SELECT** setelah klausa **WHERE** pada **Subquery** untuk menampilkan **id_project**, **tanggal_tertinggi**, **tanggal_terendah**. Dimana data yang ditampilkan memiliki kondisi **start_date** lebih dari **6 Mei 2020** ditambah dengan penggunaan **GROUP BY** untuk mengelompokkan data berdasarkan **id_project**.



Laporan Sementara

Langkah Keenambelas

Membuat subquery dengan menggunakan minimal 3 nested query.

Query

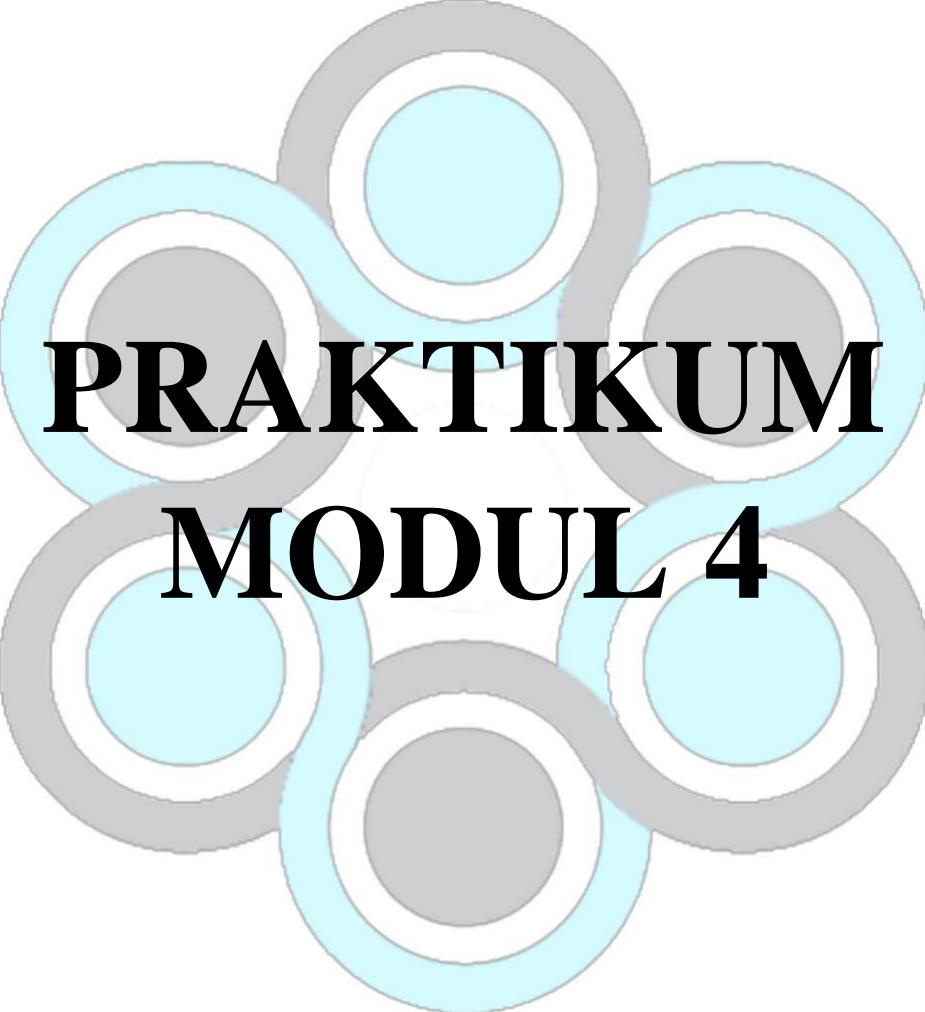
```
SELECT id_project, (SELECT SUM(id_job) FROM project WHERE id_project < (SELECT COUNT(id_project) AS jumlah_id_project FROM project)) AS akumulasi_id_job FROM project GROUP BY id_project;
```

Screenshot

```
SQL> SELECT id_project, (SELECT SUM(id_job) FROM project WHERE id_project < (SELECT COUNT(id_project) AS jumlah_id_project FROM project)) AS akumulasi_id_job FROM project GROUP BY id_project;
ID_PROJECT AKUMULASI_ID_JOB
-----
      2          2203
      5          2203
     21          2203
     22          2203
     23          2203
     24          2203
     25          2203
7 rows selected.
SQL>
```

Analisa :

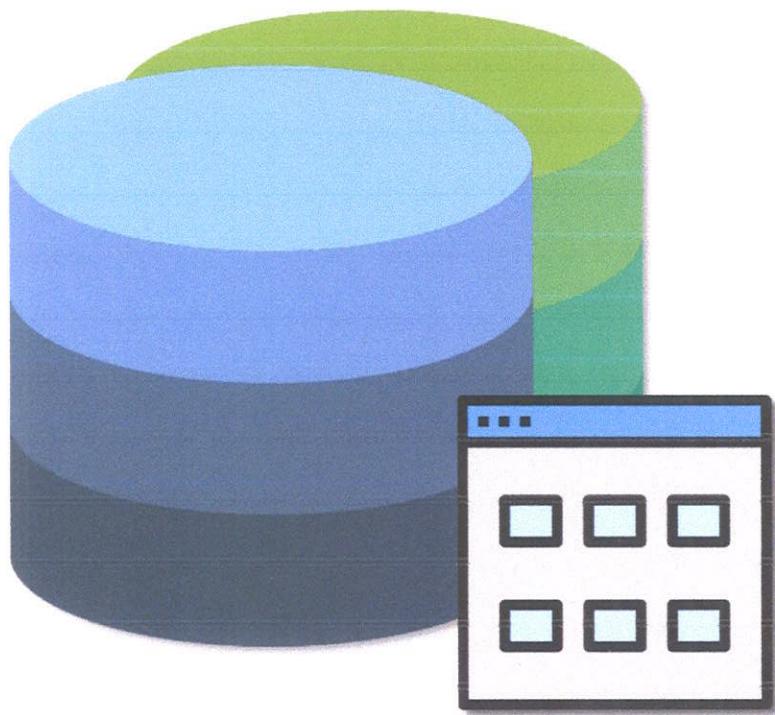
Menggunakan klausua **SELECT** dengan 3 nested query pada pembuatan **Subquery** yang digunakan untuk menampilkan data **id_project** dan **akumulasi_id_job** pada table **Project**. Dimana data yang ditampilkan memiliki kondisi dimana **id_project** harus kurang dari **jumlah_id_project**. Lalu disertai penggunaan **GROUP BY** untuk mengelompokkan data berdasarkan **id_project**.



PRAKTIKUM MODUL 4

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020

LAPORAN PRAKTIKUM

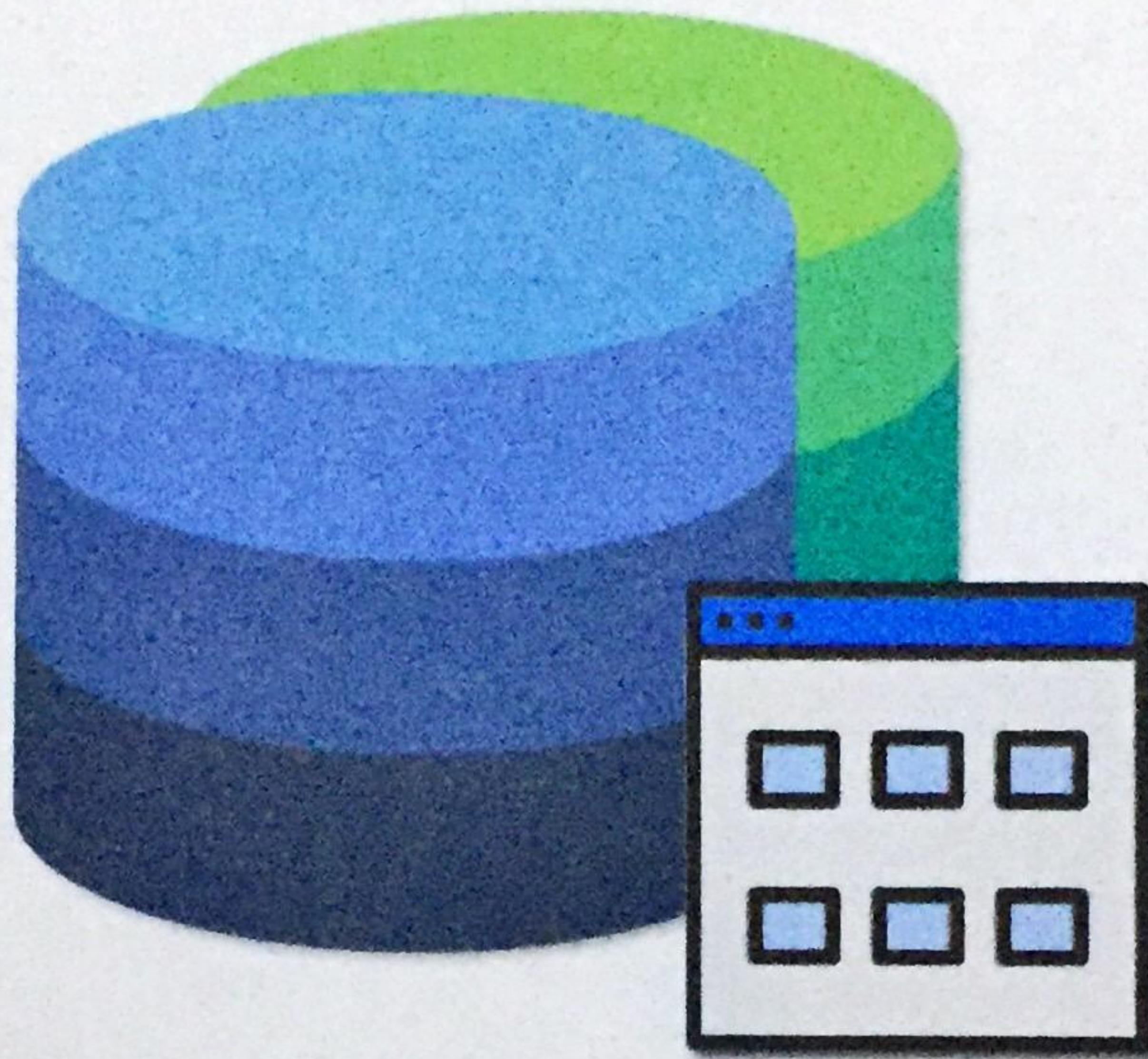


PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmad Muchlasin
NPM : 06.2018-1.06941
MODUL : 10

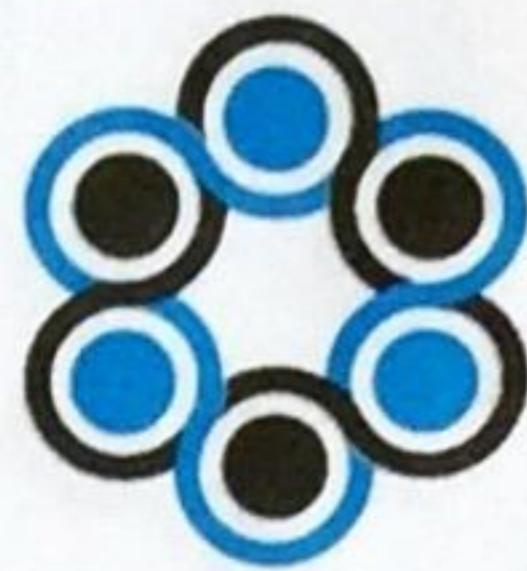
--

LAPORAN PENDAHULUAN



**PRAKTIKUM BASIS DATA
PERIODE V - Tahun 2019/2020**

NAMA : Achmad Muchlasin
NPM : 06.2018.1.06941
MODUL : IV



Tugas Pertanyaan

Modul IV

BAB II

4.1 Tujuan

1. Praktikan dapat mengetahui konsep Join
2. Praktikan dapat mengetahui fungsi view

4.2 Kriteria Penilaian

1. Praktikan dapat mengimplementasikan dan membedakan jenis join table
2. Praktikan dapat membuat view.

4.3 Soal Pendahuluan

1. Perhatikan tabel berikut :

NAMA_PRODUK	NAMA_KATEGORI	HARGA_SATUAN	STOK_PRODUK
Indomie	Makanan	2000	80
Minimas	Minuman	2000	40
Jeruk	Buah-Sayur	5000	100
Samsung	Elektronik	8.000.000	35
Supreme	Baju	35.000	95
Poti Kukus	Makanan	2500	10
Kue Blak	Makanan	5500	90
Cranberries	Makanan	2800	70
Semangka	Minuman	10.000	40
Kueku	Makanan	2500	50

Buatlah sebuah query untuk menampilkan tabel seperti dibawah !

- Cek terlebih dahulu tabel yang memiliki primary key dan foreign key yang saling terhubung .



Tugas Pertanyaan

a. Tabel Produk

SQL > DESC produk_06941 ;

Name
ID_PRODUK
ID_Pemasok
ID_KATEGORI
NAMA_PRODUK
HARGA_SATUAN
STOK_PRODUK

b. Tabel Kategori

SQL DESC kategori_06941 ;

Name
ID_KATEGORI
NAMA_KATEGORI
DESKRIPSI

- Buat Query untuk melakukan join table.

SQL > SELECT a.nama_produk, b.nama_kategori, a.harga_satuan,
a.stok_produk

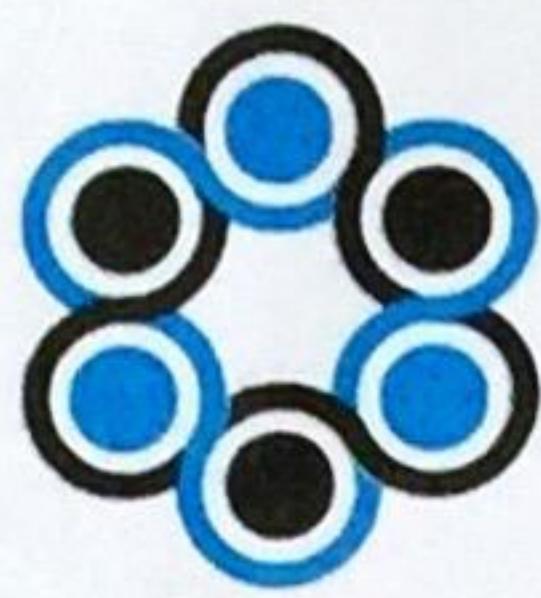
2 FROM produk_06941 a JOIN

3 kategori_06941 b

4 ON a.id_kategori = b.id_kategori

5 WHERE rownum <= 10;

- 2. Perhatikan gambar berikut:



Tugas Pertanyaan

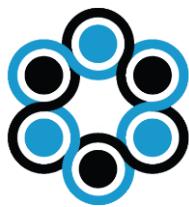
ID-PEMASOK	NAMA PERUSAHAAN	ALAMAT	NAMA - PRODUK
1	DistrIndomart	Lombok	Indomie
2	DistrAlfamart	Madura	Mammas
3	Distribgiant	Sidoarjo	Serub
4	DistrTransmart	Jupiter	Samsung
5	DistrSuprindo	Pluto	Supreme
1	DistrIndomart	Lombok	Roti Kukus
1	Distr Indomart	Lombok	Bu Balok
1	DistrIndomart	Lombok	Cranberries
2	DistrAlfamart	Madura	Semangka
5	DistrSuprindo	Pluto	kuekyu

Buatlah sebuah view untuk menampilkan table seperti diatas!

- Query untuk view.

SQL> CREATE VIEW list_remasok

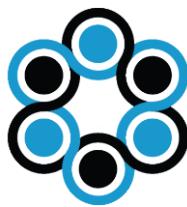
```
2 AS SELECT a.id_pemasok, a.nama_perusahaan, a.alamat, b.  
    nama_produk  
3 FROM pemasok_06941 a  
4 JOIN produk_06941 b  
5 ON a.id_pemasok = b.id_pemasok  
6 JOIN kategori_06941 c  
7 ON b.id_kategori = c.id_kategori;
```



Tugas Praktikum

Soal Praktikum

1. Insert 5 data pada setiap tabel
2. Buatlah query laporan dengan jelas sesuai proyek Anda dengan ketentuan sebagai berikut :
 - a. Equi Join minimal 2 tabel dengan menerapkan Operator Pembanding dan Klausa
 - b. Left Join minimal 3 tabel dengan menerapkan Klausa dan Subquery
 - c. Right Join minimal 3 tabel dengan menerapkan Aggregate dan Operator Logika
- Note : Tampilkan kolom pada tabel pertama dan kolom pada tabel kedua dengan memperhatikan kasus yang akan Anda laporkan.
3. Buatlah view dengan menerapkan DML.
4. Buatlah view dari masing-masing soal pada nomor 2.
5. Terapkan DML pada View pada nomor 4.



Tugas Praktikum

Langkah Pertama

Insert minimal 5 data pada table **Employee** menggunakan **INSERT ALL**.

Query

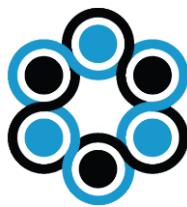
```
INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (111, 'Roni', 'Sedati', 'L',
'1111111', 'roni@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (112, 'Yuna', 'Waru', 'P', '1222222',
'yuna@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (113, 'Rainy', 'Sidoarjo', 'P',
'1333333', 'rainy@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (114, 'Moza', 'Candi', 'L', '1444444',
'moza@gmail.com')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  3  values (111, 'Roni', 'Sedati', 'L', '1111111', 'roni@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  5  values (112, 'Yuna', 'Waru', 'P', '1222222', 'yuna@gmail.com')
  6  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  7  values (113, 'Rainy', 'Sidoarjo', 'P', '1333333', 'rainy@gmail.com')
  8  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  9  values (114, 'Moza', 'Candi', 'L', '1444444', 'moza@gmail.com')
10  SELECT 1 FROM dual;
11
5 rows created.
```

Analisa

Insert dilakukan pada table **Employee** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Praktikum

Langkah Kedua

Insert minimal 5 data pada table **Client** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (21, 'Langitnet', 'Sidoarjo',
'admin@langit.net', '12111')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (22, 'PT PAL', 'Tanjung Perak',
'hrd@pal.co.id', '12222')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (23, 'PT Semen Indonesia', 'Gresik',
'inquiry@semen.co.id', '12333')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (24, 'PT DOK Surabaya', 'Tanjung
Perak', 'admin.hrd@dok.co.id', '12444')
SELECT 1 FROM dual;
```

Screenshot

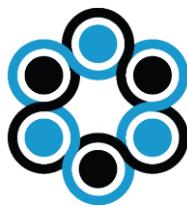
```
SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  6, 'Foxtrot Alto', 'Bukti Darmo', 'hr.g@foxalto.co', '14444')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  7, 'Padinet', 'Mayjen Sungkono', 'admin.ho@padi.net', '15555')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  8, 'Grand Tower Corp', 'Basuki Rahmat', 'admin.hr@gtower.co.id', '16666')
  5  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  9, 'PT Sejahtera Abadi', 'Tanggulangin Sda', 'inquiry@sa.co.id', '17777')
  6  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (2
  0, 'Maersk Corp', 'Pandaan', 'hr.ho@maersk.co.id', '18888')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```

Analisa :

Insert dilakukan pada table **Client** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Praktikum

Langkah Ketiga

Insert minimal 5 data pada table **Job** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO job (id_job, id_client, job_name,
category, description) values (1111, 21,
'Admin IT', 'IT', 'Input ticket and generate
ticket report')
INTO job (id_job, id_client, job_name,
category, description) values (1112, 22, 'GA
Drafter', 'Engineer', 'Collect and submit
tender')
INTO job (id_job, id_client, job_name,
category, description) values (1113, 21, 'IT
Support', 'IT', 'Maintain network
peripheral')
INTO job (id_job, id_client, job_name,
category, description) values (1114, 25,
'General Affair', 'General', 'Manage office
tools')
INTO job (id_job, id_client, job_name,
category, description) values (1115, 23,
'Admin HRD', 'HR', 'Input data employee')
SELECT 1 FROM dual;
```

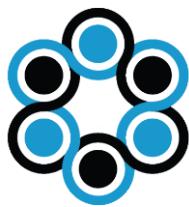
Screenshot

```
SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1111, 21, 'A
dmin IT', 'IT', 'Input ticket and generate ticket report')
  3  INTO job (id_job, id_client, job_name, category, description) values (1112, 22, 'G
A Drafter', 'Engineer', 'Collect and submit tender')
  4  INTO job (id_job, id_client, job_name, category, description) values (1113, 21, 'I
T Support', 'IT', 'Maintain network peripheral')
  5  INTO job (id_job, id_client, job_name, category, description) values (1114, 25, 'G
eneral Affair', 'General', 'Manage office tools')
  6  INTO job (id_job, id_client, job_name, category, description) values (1115, 23, 'A
dmn HRD', 'HR', 'Input data employee')
  7  SELECT 1 FROM dual;

5 rows created.
```

Analisa :

Insert dilakukan pada table **Job** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Praktikum

Langkah Keempat

Insert minimal 5 data pada table **Project** menggunakan **INSERT INTO**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1111, 111,
to_date('01/06/2020', 'dd/mm/yyyy'),
to_date('30/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1112, 112,
to_date('02/06/2020', 'dd/mm/yyyy'),
to_date('10/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1113, 113,
to_date('10/06/2020', 'dd/mm/yyyy'),
to_date('30/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1114, 112,
to_date('15/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1111, 111, to_date('01/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1112, 112, to_date('02/06/2020', 'dd/mm/yyyy'), to_date('10/06/2020', 'dd/mm/yyyy'));
1 row created.

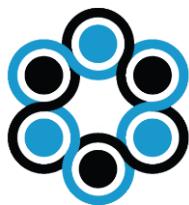
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1113, 113, to_date('10/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1114, 112, to_date('15/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL>
```

Analisa :

Insert dilakukan pada table **Project** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Tugas Praktikum

Langkah Kelima

Membuat query dengan menerapkan Equi Join minimal 2 tabel dengan menerapkan Operator Pembanding dan Klausuquery.

Query

```
select j.id_job, j.job_name, j.description,
c.client_name from job j inner join client c
on j.id_client = c.id_client
where j.id_job <= 1110 order by j.id_job;
```

Screenshot

ID_JOB	JOB_NAME	DESCRIPTION	CLIENT_NAME
1181 Network Administrator	Network Monitoring	Design and monitoring network	Maersk Corp
1182 Front-End Developer	Design and develop UI Website	Design and develop UI Website	Ikalye Vessel
1183 Vessel Manager	Managing Vessel	Managing Vessel	Golden Lineyard
1184 Accounting Staff	Reviewing Financial Statement	Reviewing Financial Statement	Oreile Shipyard
1185 QA/QC Vessel	Inspect and test materials	Inspect and test materials	Orisca Lines
1186 Quality Control Officer	Inspect and test materials	Inspect and test materials	Grand Tower Corp
1187 Business Dev	Collect and submit tender	Collect and submit tender	CCCL
1188 HR Support	Manage employee peripheral	Manage employee peripheral	Maersk Corp
1189 HRD	Manage employee	Manage employee	
1190 General Affairs	Manage office tools	Manage office tools	

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Equi Join dengan jenis Inner Join. Tabel yang digunakan untuk query di atas adalah Table **Job** dan Table **Client**. Tujuan query di atas adalah menampilkan kolom id_job, job_name, job_description, dan client_name.



Tugas Praktikum

Langkah Keenam

Membuat query dengan menerapkan Left Join minimal 3 tabel dengan menerapkan Klausa dan Subquery.

Query

```
select p.id_project, j.job_name,
e.employee_name, p.start_date, p.end_date
from project p left join job j
on p.id_job = j.id_job left join employee e
on p.id_employee = e.id_employee
where p.id_project in (select id_project from
project where start_date <=
to_date('10/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> select p.id_project, j.job_name, e.employee_name, p.start_date, p.end_date from project p left join job j
  2  on p.id_job = j.id_job left join employee e
  3  on p.id_employee = e.id_employee
  4  where p.id_project in (select id_project from project where start_date <= to_date('10/06/2020', 'dd/mm/yyyy'));

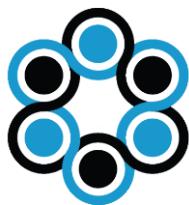
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
-----  -----          -----
21 Software Engineer        Maya             06-MAY-20 15-MAY-20
22 Bussiness Dev           Aga              07-MAY-20 18-MAY-20
23 IT Support               Nara             08-MAY-20 30-MAY-20
24 HRD                      Aga              09-MAY-20 21-MAY-20
25 General Affair          Tara             10-MAY-20 24-MAY-20
41 Admin IT                 Roni             01-JUN-20 30-JUN-20
42 GA Drafter               Yuna             02-JUN-20 10-JUN-20
43 IT Support                Rainy            10-JUN-20 30-JUN-20
2 Front-End Developer       Sterling         02-MAY-20 10-MAY-20
5 Network Administrator     Sterling         07-MAY-20 28-MAY-20

10 rows selected.

SQL>
```

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Left Join dengan menerapkan Klausa dan Subquery pada 3 table yang terlibat. Tabel yang digunakan untuk query di atas adalah **Project, Job, Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, employee_name, start_date, end_date.



Tugas Praktikum

Langkah Ketujuh

Membuat query dengan menerapkan Right Join minimal 3 tabel dengan menerapkan Aggregate dan Operator Logika.

Query

```
select p.id_project, j.job_name,
j.description, e.employee_name, e.email from project p right join job j
on p.id_job = j.id_job right join employee e
on p.id_employee = e.id_employee
where p.id_project > (select
count(id_project) from project) AND
p.end_date > to_date('30/05/2020',
'dd/mm/yyyy');
```

Screenshot

ID_PROJECT	NAME	DESCRIPTION	EMPLOYEE_NAME	EMAIL
41 Admin IT	Input ticket and generate ticket report		Roni	roni@gmail.com
42 Admin Support	Maintain network peripheral		Roni	roni@gmail.com
43 IT Support	Maintain network peripheral		Rainy	rainy@gmail.com
44 Admin HR	Input data employee		Rainy	rainy@gmail.com
45 Admin HRD			Pola	pola@gmail.com

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Right Join dengan menerapkan Agregate dan Operator Pembanding pada 3 table yang terlibat. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, description, employee_name, email.



Tugas Praktikum

Langkah Kedelapan

Membuat view dengan menerapkan DML.

Query

```
CREATE OR REPLACE VIEW CLIENT_EXISTING_V AS
SELECT id_client, client_name, client_addr,
client_email FROM client;
```

Screenshot

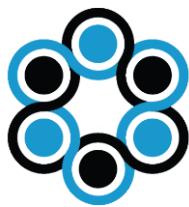
```
SQL> CREATE OR REPLACE VIEW CLIENT_EXISTING_V AS SELECT id_client, client_name, client_addr, client_email FROM client;
View created.

SQL> SELECT * FROM CLIENT_EXISTING_V
  2  ;
ID_CLIENT CLIENT_NAME          CLIENT_ADDR           CLIENT_EMAIL
-----  -----
16 Footrot Alto             Buttil Darmo
17 Padinet                  Hayjen Sungkono
18 PT. Sinar Super Corp    Basuki Rahmat
19 PT. Sejahtera Abadi     Tegalalangin Sda
20 Huarsik Corp            Pandan
21 IstimewaNet
22 PT Pali                  Tanjung Perak
23 PT Sesmen Indonesia     Gresik
24 PT. SIK Surabaya         Tambang Perak
25 D-Net                    Basuki Rahmat
11 Hexamedika Corp          Rungkut Industri Surabaya
                               hr.gm@roxato.co.id
                               admin.hc@adi.net
                               support@hexa.co.id
                               hr.hg@huarsik.co.id
                               admin.hi@huarsik.net
                               hr.dg@pali.co.id
                               inquiry@sesmen.co.id
                               sekretaris@sesmen.co.id
                               l1@t.net.co.id
                               admin.h@hexa.co.id
IS rows selected.

SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW sederhana untuk memunculkan kolom id_client, client_name, client_addr, client_email dari table **Client** dengan nama view **CLIENT_EXISTING_V**. Penamaan dengan sengaja diberi tambahan “V” agar mudah mengetahui bahwa nama tersebut adalah view yang kita buat.



Tugas Praktikum

Langkah Kesembilan

Membuat view dari masing-masing soal pada nomor 2a.

Query

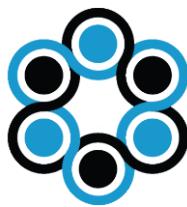
```
CREATE OR REPLACE VIEW JOB_FROM_CLIENT_V AS
select j.id_job, j.job_name, j.description,
c.id_client, c.client_name from job j inner
join client c on j.id_client = c.id_client
where j.id_job <= 1110 order by j.id_job;
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW JOB_FROM_CLIENT_V AS select j.id_job, j.job_name, j.description, c.id_client, c.client_na
me from job j inner join client c on j.id_client = c.id_client
2 where j.id_job <= 1110 order by j.id_job;
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Equi Join dengan jenis Inner Join dengan nama **JOB_FROM_CLIENT_V**. Tabel yang digunakan untuk query di atas adalah Table **Job** dan Table **Client**. Tujuan query di atas adalah menampilkan kolom id_job, job_name, job_description, dan client_name.



Tugas Praktikum

Langkah Kesepuluh

Membuat view dari masing-masing soal pada nomor 2b.

Query

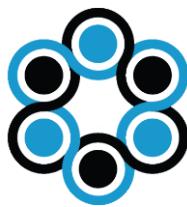
```
CREATE OR REPLACE VIEW PROJECT_UNDER_10JUNE_V
AS
select p.id_project, j.job_name,
e.employee_name, p.start_date, p.end_date
from project p left join job j
on p.id_job = j.id_job left join employee e
on p.id_employee = e.id_employee
where p.id_project in (select id_project from
project where start_date <=
to_date('10/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW PROJECT_UNDER_10JUNE_V AS
  2  select p.id_project, j.job_name, e.employee_name, p.start_date, p.end_date from project p left join j
ob j
  3  on p.id_job = j.id_job left join employee e
  4  on p.id_employee = e.id_employee
  5  where p.id_project in (select id_project from project where start_date <= to_date('10/06/2020', 'dd/m
m/yyyy'));
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Left Join pada 3 table yang terlibat dengan nama **PROJECT_UNDER_10JUNE_V**. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom `id_project`, `job_name`, `employee_name`, `start_date`, `end_date`.



Tugas Praktikum

Langkah Kesebelas

Membuat view dari masing-masing soal pada nomor 2c.

Query

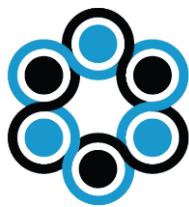
```
CREATE OR REPLACE VIEW DUE_DATE_AFTER_30MAY
AS
select p.id_project, j.job_name,
j.description, e.employee_name, e.email from
project p right join job j
on p.id_job = j.id_job right join employee e
on p.id_employee = e.id_employee
where p.id_project > (select
count(id_project) from project) AND
p.end_date > to_date('30/05/2020',
'dd/mm/yyyy');
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW DUE_DATE_AFTER_30MAY AS
  2  select p.id_project, j.job_name, j.description, e.employee_name, e.email from project p right join jo
  b j
  3  on p.id_job = j.id_job right join employee e
  4  on p.id_employee = e.id_employee
  5  where p.id_project > (select count(id_project) from project) AND p.end_date > to_date('30/05/2020',
dd/mm/yyyy');
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Right Join pada 3 table yang terlibat dengan nama **DUE_DATE_AFTER_30MAY**. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, description, employee_name, email.



Tugas Praktikum

Langkah Keduabelas

Menapkan perintah DML pada View nomer 4a.

Query

```
UPDATE JOB_FROM_CLIENT_V
SET job_name = 'NOC'
WHERE id_client = 11;
```

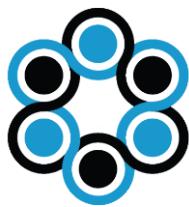
Screenshot

```
SQL> UPDATE JOB_FROM_CLIENT_V
2 SET job_name = 'NOC'
3 WHERE id_client = 11;
1 row updated.

SQL> SELECT * FROM PROJECT_UNDER_10JUNE_V;
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
----- -----------
    21 Software Engineer      Maya              06-MAY-20 15-MAY-20
    24 HRD                   Aga               09-MAY-20 20-MAY-20
    23 Business Dev          Aga               07-MAY-20 18-MAY-20
    25 IT Support             Nara              08-MAY-20 30-MAY-20
    41 Admin IT               Tira              10-MAY-20 24-MAY-20
    42 GA Drafter             Roni              01-JUN-20 30-JUN-20
    43 IT Support             Rainy             02-JUN-20 30-JUN-20
    2 Front-End Developer     Sterling          02-MAY-20 10-MAY-20
    5 NOC                    Sterling          07-MAY-20 28-MAY-20
10 rows selected.
```

Analisa :

Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **update**. Data yang diupdate disini adalah **job_name** yang awalnya **Network Administrator** menjadi **NOC** dengan mengacu kondisi pada klausa where pada view **JOB_FROM_CLIENT_V**.



Tugas Praktikum

Langkah Ketigabelas

Menapkan perintah DML pada View nomer 4b.

Query

```
DELETE FROM PROJECT_UNDER_10JUNE_V
WHERE job_name = 'HRD';
```

Screenshot

```
SQL> DELETE FROM PROJECT_UNDER_10JUNE_V
  2 WHERE job_name = 'HRD';
1 row deleted.

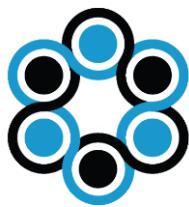
SQL> SELECT * FROM PROJECT_UNDER_10JUNE_V;
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
----- ---------
 21 Software Engineer        Maya
 22 Bussiness Dev           Aga
 23 IT Support               Nara
 25 General Affair          Tara
 41 Admin                    Roni
 42 GA Drafter               Yuna
 43 IT Support                Maily
  2 Front-End Developer      Sterling
  5 NOC                      Sterling

9 rows selected.

SQL>
```

Analisa :

Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **delete**. Data yang didelete disini adalah job_name = 'HRD' dengan mengacu kondisi pada klausula where pada view **PROJECT_UNDER_10JUNE_V**.



Tugas Praktikum

Langkah Keempatbelas

Menapkan perintah DML pada View nomer 4c.

Query

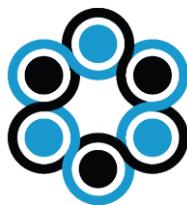
```
SELECT * FROM DUE_DATE_AFTER_30MAY ORDER BY  
id_project DESC;
```

Screenshot

```
SQL> SELECT * FROM DUE_DATE_AFTER_30MAY ORDER BY id_project DESC;  
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      DESCRIPTION           EMAIL  
-----  
45 Admin HRD                Moza              Input data employee    moza@gmail.com  
44 General Affair           Yuna              Manage office tools  yuna@gmail.com  
43 IT Support                Rainy             Maintain network peripheral  rainy@gmail.com  
42 GA Drafter               Yuna              Collect and submit tender  yuna@gmail.com  
41 Admin IT                 Roni              Input ticket and generate ticket report  roni@gmail.com  
SQL>
```

Analisa :

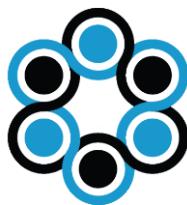
Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **select**. Data yang ditampilkan disini adalah seluruh data dari view yang sudah kita buat yakni **DUE_DATE_AFTER_30MY** dengan tambahan penerapan fungsi **ORDER BY DESC** pada **id_project**.



Laporan Sementara

Soal Praktikum

1. Insert 5 data pada setiap tabel
2. Buatlah query laporan dengan jelas sesuai proyek Anda dengan ketentuan sebagai berikut :
 - a. Equi Join minimal 2 tabel dengan menerapkan Operator Pembanding dan Klausa
 - b. Left Join minimal 3 tabel dengan menerapkan Klausa dan Subquery
 - c. Right Join minimal 3 tabel dengan menerapkan Aggregate dan Operator Logika
- Note : Tampilkan kolom pada tabel pertama dan kolom pada tabel kedua dengan memperhatikan kasus yang akan Anda laporkan.
3. Buatlah view dengan menerapkan DML.
4. Buatlah view dari masing-masing soal pada nomor 2.
5. Terapkan DML pada View pada nomor 4.



Laporan Sementara

Langkah Pertama

Insert minimal 5 data pada table **Employee** menggunakan **INSERT ALL**.

Query

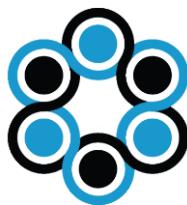
```
INSERT ALL
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (111, 'Roni', 'Sedati', 'L',
'1111111', 'roni@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (112, 'Yuna', 'Waru', 'P', '1222222',
'yuna@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (113, 'Rainy', 'Sidoarjo', 'P',
'1333333', 'rainy@gmail.com')
INTO employee (id_employee, employee_name,
employee_addr, gender, phone_no, email)
values (114, 'Moza', 'Candi', 'L', '1444444',
'moza@gmail.com')
SELECT 1 FROM dual;
```

Screenshot

```
SQL> INSERT ALL
  2  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  3  values (111, 'Roni', 'Sedati', 'L', '1111111', 'roni@gmail.com')
  4  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  5  values (112, 'Yuna', 'Waru', 'P', '1222222', 'yuna@gmail.com')
  6  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  7  values (113, 'Rainy', 'Sidoarjo', 'P', '1333333', 'rainy@gmail.com')
  8  INTO employee (id_employee, employee_name, employee_addr, gender, phone_no, email)
  9  values (114, 'Moza', 'Candi', 'L', '1444444', 'moza@gmail.com')
10  SELECT 1 FROM dual;
11
5 rows created.
```

Analisa

Insert dilakukan pada table **Employee** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Laporan Sementara

Langkah Kedua

Insert minimal 5 data pada table **Client** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (21, 'Langitnet', 'Sidoarjo',
'admin@langit.net', '12111')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (22, 'PT PAL', 'Tanjung Perak',
'hrd@pal.co.id', '12222')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (23, 'PT Semen Indonesia', 'Gresik',
'inquiry@semen.co.id', '12333')
INTO client (id_client, client_name,
client_addr, client_email, client_phone_no)
values (24, 'PT DOK Surabaya', 'Tanjung
Perak', 'admin.hrd@dok.co.id', '12444')
SELECT 1 FROM dual;
```

Screenshot

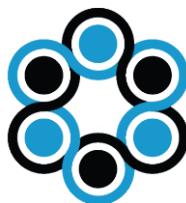
```
SQL> INSERT ALL
  2  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  6, 'Foxtrot Alto', 'Bukti Darmo', 'hr.g@foxalto.co', '14444')
  3  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  7, 'Padinet', 'Mayjen Sungkono', 'admin.h@padi.net', '15555')
  4  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  8, 'Grand Tower Corp', 'Basuki Rahmat', 'admin.hr@gtower.co.id', '16666')
  5  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (1
  9, 'PT Sejahtera Abadi', 'Tanggulangin Sda', 'inquiry@sa.co.id', '17777')
  6  INTO client (id_client, client_name, client_addr, client_email, client_phone_no) values (2
  0, 'Maersk Corp', 'Pandaan', 'hr.ho@maersk.co.id', '18888')
  7  SELECT 1 FROM dual;

5 rows created.

SQL>
```

Analisa :

Insert dilakukan pada table **Client** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Laporan Sementara

Langkah Ketiga

Insert minimal 5 data pada table **Job** menggunakan **INSERT ALL**.

Query

```
INSERT ALL
INTO job (id_job, id_client, job_name,
category, description) values (1111, 21,
'Admin IT', 'IT', 'Input ticket and generate
ticket report')
INTO job (id_job, id_client, job_name,
category, description) values (1112, 22, 'GA
Drafter', 'Engineer', 'Collect and submit
tender')
INTO job (id_job, id_client, job_name,
category, description) values (1113, 21, 'IT
Support', 'IT', 'Maintain network
peripheral')
INTO job (id_job, id_client, job_name,
category, description) values (1114, 25,
'General Affair', 'General', 'Manage office
tools')
INTO job (id_job, id_client, job_name,
category, description) values (1115, 23,
'Admin HRD', 'HR', 'Input data employee')
SELECT 1 FROM dual;
```

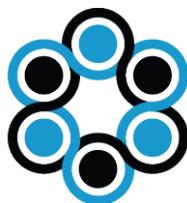
Screenshot

```
SQL> INSERT ALL
  2  INTO job (id_job, id_client, job_name, category, description) values (1111, 21, 'A
dmin IT', 'IT', 'Input ticket and generate ticket report')
  3  INTO job (id_job, id_client, job_name, category, description) values (1112, 22, 'G
A Drafter', 'Engineer', 'Collect and submit tender')
  4  INTO job (id_job, id_client, job_name, category, description) values (1113, 21, 'I
T Support', 'IT', 'Maintain network peripheral')
  5  INTO job (id_job, id_client, job_name, category, description) values (1114, 25, 'G
eneral Affair', 'General', 'Manage office tools')
  6  INTO job (id_job, id_client, job_name, category, description) values (1115, 23, 'A
dmn HRD', 'HR', 'Input data employee')
  7  SELECT 1 FROM dual;

5 rows created.
```

Analisa :

Insert dilakukan pada table **Job** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Laporan Sementara

Langkah Keempat

Insert minimal 5 data pada table **Project** menggunakan **INSERT INTO**.

Query

```
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1111, 111,
to_date('01/06/2020', 'dd/mm/yyyy'),
to_date('30/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1112, 112,
to_date('02/06/2020', 'dd/mm/yyyy'),
to_date('10/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1113, 113,
to_date('10/06/2020', 'dd/mm/yyyy'),
to_date('30/06/2020', 'dd/mm/yyyy'));
INSERT INTO project (id_project, id_job,
id_employee, start_date, end_date) values
(id_project.nextval, 1114, 112,
to_date('15/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1111, 111, to_date('01/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1112, 112, to_date('02/06/2020', 'dd/mm/yyyy'), to_date('10/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1113, 113, to_date('10/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

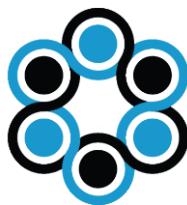
SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1114, 112, to_date('15/06/2020', 'dd/mm/yyyy'), to_date('30/06/2020', 'dd/mm/yyyy'));
1 row created.

SQL> INSERT INTO project (id_project, id_job, id_employee, start_date, end_date) values
(id_project.nextval, 1115, 114, to_date('15/06/2020', 'dd/mm/yyyy'), to_date('15/07/2020', 'dd/mm/yyyy'));
1 row created.

SQL>
```

Analisa :

Insert dilakukan pada table **Project** dengan menggunakan query **INSERT ALL** agar beberapa data yang ingin dimasukkan bisa di insert dalam 1 line query dan 1 kali eksekusi.



Laporan Sementara

Langkah Kelima

Membuat query dengan menerapkan Equi Join minimal 2 tabel dengan menerapkan Operator Pembanding dan Klausuquery.

Query

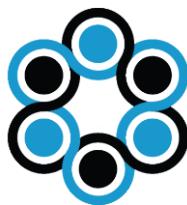
```
select j.id_job, j.job_name, j.description,
c.client_name from job j inner join client c
on j.id_client = c.id_client
where j.id_job <= 1110 order by j.id_job;
```

Screenshot

ID_JOB	JOB_NAME	DESCRIPTION	CLIENT_NAME
1181 Network Administrator		Design and monitoring network	Havelsk Corp
1182 Front-End Developer		Design and develop UI Website	Iekalye Vessel
1183 Tester		Testing software	Golden Unipred
1184 Accounting Staff		Reviewing financial statement	Oreila Shipyard
1185 QA/QC Vessel		Inspect and test materials	Orisca Corp
1186 Project Manager		Planning and managing project	Ortis Corp
1187 Business Dev		Collect and submit tender	Grand Tower Corp
1188 HR Support		Managing HR & peripheral	Orbis Corp
1189 HRD		Manage employee	Havelsk Corp
1190 General Affairs		Manage office tools	Havelsk Corp
10 rows selected.			
SQLs			

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Equi Join dengan jenis Inner Join. Tabel yang digunakan untuk query di atas adalah Table **Job** dan Table **Client**. Tujuan query di atas adalah menampilkan kolom id_job, job_name, job_description, dan client_name.



Laporan Sementara

Langkah Keenam

Membuat query dengan menerapkan Left Join minimal 3 tabel dengan menerapkan Klausa dan Subquery.

Query

```
select p.id_project, j.job_name,
e.employee_name, p.start_date, p.end_date
from project p left join job j
on p.id_job = j.id_job left join employee e
on p.id_employee = e.id_employee
where p.id_project in (select id_project from
project where start_date <=
to_date('10/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> select p.id_project, j.job_name, e.employee_name, p.start_date, p.end_date from project p left join job j
  2  on p.id_job = j.id_job left join employee e
  3  on p.id_employee = e.id_employee
  4  where p.id_project in (select id_project from project where start_date <= to_date('10/06/2020', 'dd/mm/yyyy'));

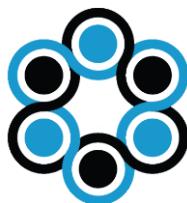
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
-----  -----          -----          -----
21 Software Engineer        Maya             06-MAY-20 15-MAY-20
22 Bussiness Dev           Aga              07-MAY-20 18-MAY-20
23 IT Support               Nara             08-MAY-20 30-MAY-20
24 HRD                      Aga              09-MAY-20 21-MAY-20
25 General Affair          Tara             10-MAY-20 24-MAY-20
41 Admin IT                 Roni             01-JUN-20 30-JUN-20
42 GA Drafter               Yuna             02-JUN-20 10-JUN-20
43 IT Support                Rainy            10-JUN-20 30-JUN-20
2 Front-End Developer       Sterling         02-MAY-20 18-MAY-20
5 Network Administrator     Sterling         07-MAY-20 28-MAY-20

10 rows selected.

SQL>
```

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Left Join dengan menerapkan Klausa dan Subquery pada 3 table yang terlibat. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom `id_project`, `job_name`, `employee_name`, `start_date`, `end_date`.



Laporan Sementara

Langkah Ketujuh

Membuat query dengan menerapkan Right Join minimal 3 tabel dengan menerapkan Aggregate dan Operator Logika.

Query

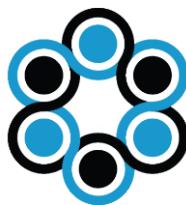
```
select p.id_project, j.job_name,
j.description, e.employee_name, e.email from
project p right join job j
on p.id_job = j.id_job right join employee e
on p.id_employee = e.id_employee
where p.id_project > (select
count(id_project) from project) AND
p.end_date > to_date('30/05/2020',
'dd/mm/yyyy');
```

Screenshot

ID_PROJECT_NAM	DESCRIPTION	EMPLOYEE_NAM	EMAIL
41 Admin IT	Input ticket and generate ticket report	Roni	roni@gmail.com
42 Admin Support	Maintain network peripheral	Roni	roni@gmail.com
43 IT Support	Maintain network peripheral	Rainy	rainy@gmail.com
44 Admin HR	Input data employee	Rainy	rainy@gmail.com
45 Admin HRD	Input data employee	Pola	pola@gmail.com

Analisa :

Perintah di atas berfungsi untuk membuat query dengan menerapkan Right Join dengan menerapkan Agregate dan Operator Pembanding pada 3 table yang terlibat. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, description, employee_name, email.



Laporan Sementara

Langkah Kedelapan

Membuat view dengan menerapkan DML.

Query

```
CREATE OR REPLACE VIEW CLIENT_EXISTING_V AS  
SELECT id_client, client_name, client_addr,  
client_email FROM client;
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW CLIENT_EXISTING_V AS SELECT id_client, client_name, client_addr, client_email FROM client;  
View created.  
SQL> SELECT * FROM CLIENT_EXISTING_V  
2 ;  


| ID_CLIENT | CLIENT_NAME          | CLIENT_ADDR               | CLIENT_EMAIL            |
|-----------|----------------------|---------------------------|-------------------------|
| 16        | Footrot Alto         | Bukti Darro               | hr.g@foxato.co          |
| 17        | Padinet              | Hayjen Sungkono           | admin.hopadi.net        |
| 18        | PT. Hauer Corp       | Basko Sidoarjo            | admin@hauercorp.co.id   |
| 19        | PT. Sejahtera Abadi  | Tanggalingin Sda          | inquiry@sejaha.co.id    |
| 20        | Huarsik Corp         | Pandan                    | hr.huarsik.co.id        |
| 21        | Siapnet              | Sidoarjo                  | admin.siapnet.net       |
| 22        | PT PIA               | Tanjung Perak             | hr@piaindonesia.co.id   |
| 23        | PT. Seamen Indonesia | Gresik                    | inquiry@seamen.co.id    |
| 24        | PT. PK Surabaya      | Tanjung Perak             | admin@pk-surabaya.co.id |
| 25        | D-Net                | Basuki Rahmat             | l1@dn-net.co.id         |
| 11        | Hexamedika Corp      | Rungkit Industri Surabaya | admin.h@hexa.co.id      |
| ID_CLIENT | CLIENT_NAME          | CLIENT_ADDR               | CLIENT_EMAIL            |
| 12        | Ekalya Vessel        | Darso Surabaya            | hr@ekalya.co.id         |
| 13        | CV Avodamitra        | Maru Sidoarjo             | hr@avodamitra.co.id     |
| 14        | PT. Orela            | Jambatan Solo             | hr@orela.co.id          |
| 15        | Orela Shipyard       | Pangkah Gresik            | support@orela.co.id     |



IS rows selected.

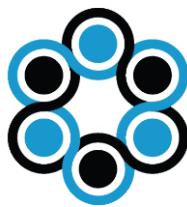


SQL>


```

Analisa :

Query di atas digunakan untuk membuat VIEW sederhana untuk memunculkan kolom id_client, client_name, client_addr, client_email dari table **Client** dengan nama view **CLIENT_EXISTING_V**. Penamaan dengan sengaja diberi tambahan “V” agar mudah mengetahui bahwa nama tersebut adalah view yang kita buat.



Laporan Sementara

Langkah Kesembilan

Membuat view dari masing-masing soal pada nomor 2a.

Query

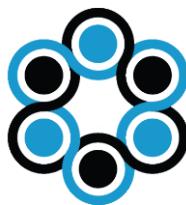
```
CREATE OR REPLACE VIEW JOB_FROM_CLIENT_V AS
select j.id_job, j.job_name, j.description,
c.id_client, c.client_name from job j inner
join client c on j.id_client = c.id_client
where j.id_job <= 1110 order by j.id_job;
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW JOB_FROM_CLIENT_V AS select j.id_job, j.job_name, j.description, c.id_client, c.client_na
me from job j inner join client c on j.id_client = c.id_client
2 where j.id_job <= 1110 order by j.id_job;
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Equi Join dengan jenis Inner Join dengan nama **JOB_FROM_CLIENT_V**. Tabel yang digunakan untuk query di atas adalah Table **Job** dan Table **Client**. Tujuan query di atas adalah menampilkan kolom **id_job**, **job_name**, **job_description**, dan **client_name**.



Laporan Sementara

Langkah Kesepuluh

Membuat view dari masing-masing soal pada nomor 2b.

Query

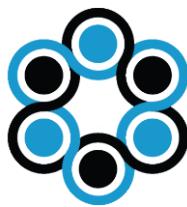
```
CREATE OR REPLACE VIEW PROJECT_UNDER_10JUNE_V
AS
select p.id_project, j.job_name,
e.employee_name, p.start_date, p.end_date
from project p left join job j
on p.id_job = j.id_job left join employee e
on p.id_employee = e.id_employee
where p.id_project in (select id_project from
project where start_date <=
to_date('10/06/2020', 'dd/mm/yyyy'));
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW PROJECT_UNDER_10JUNE_V AS
  2  select p.id_project, j.job_name, e.employee_name, p.start_date, p.end_date from project p left join j
ob j
  3  on p.id_job = j.id_job left join employee e
  4  on p.id_employee = e.id_employee
  5  where p.id_project in (select id_project from project where start_date <= to_date('10/06/2020', 'dd/m
m/yyyy'));
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Left Join pada 3 table yang terlibat dengan nama **PROJECT_UNDER_10JUNE_V**. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom **id_project**, **job_name**, **employee_name**, **start_date**, **end_date**.



Laporan Sementara

Langkah Kesebelas

Membuat view dari masing-masing soal pada nomor 2c.

Query

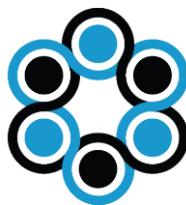
```
CREATE OR REPLACE VIEW DUE_DATE_AFTER_30MAY
AS
select p.id_project, j.job_name,
j.description, e.employee_name, e.email from
project p right join job j
on p.id_job = j.id_job right join employee e
on p.id_employee = e.id_employee
where p.id_project > (select
count(id_project) from project) AND
p.end_date > to_date('30/05/2020',
'dd/mm/yyyy');
```

Screenshot

```
SQL> CREATE OR REPLACE VIEW DUE_DATE_AFTER_30MAY AS
  2  select p.id_project, j.job_name, j.description, e.employee_name, e.email from project p right join jo
  b j
  3  on p.id_job = j.id_job right join employee e
  4  on p.id_employee = e.id_employee
  5  where p.id_project > (select count(id_project) from project) AND p.end_date > to_date('30/05/2020',
  dd/mm/yyyy');
View created.
SQL>
```

Analisa :

Query di atas digunakan untuk membuat VIEW dengan menerapkan Right Join pada 3 table yang terlibat dengan nama **DUE_DATE_AFTER_30MAY**. Tabel yang digunakan untuk query di atas adalah **Project**, **Job**, **Employee**. Tujuan query di atas adalah menampilkan kolom id_project, job_name, description, employee_name, email.



Laporan Sementara

Langkah Keduabelas

Menapkan perintah DML pada View nomer 4a.

Query

```
UPDATE JOB_FROM_CLIENT_V
SET job_name = 'NOC'
WHERE id_client = 11;
```

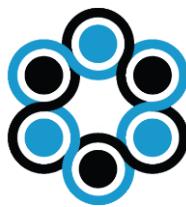
Screenshot

```
SQL> UPDATE JOB_FROM_CLIENT_V
2 SET job_name = 'NOC'
3 WHERE id_client = 11;
1 row updated.

SQL> SELECT * FROM PROJECT_UNDER_10JUNE_V;
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
----- ---------
    21 Software Engineer      Maya              06-MAY-20 15-MAY-20
    24 HRD                   Aga               09-MAY-20 28-MAY-20
    22 Business Dev          Aga               07-MAY-20 18-MAY-20
    23 IT Support             Mina              08-MAY-20 30-MAY-20
    25 General Affair        Tara              10-MAY-20 24-MAY-20
    41 Admin IT               Roni              01-JUN-20 30-JUN-20
    42 GA Draften             Yuna              02-JUN-20 10-JUN-20
    43 IT Support             Rainy             10-JUN-20 30-JUN-20
    2 Front-End Developer     Sterling          02-MAY-20 10-MAY-20
    5 NOC                    Sterling          07-MAY-20 28-MAY-20
10 rows selected.
```

Analisa :

Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **update**. Data yang diupdate disini adalah job_name yang awalnya **Network Administrator** menjadi **NOC** dengan mengacu kondisi pada klausa where pada view **JOB_FROM_CLIENT_V**.



Laporan Sementara

Langkah Ketigabelas

Menapkan perintah DML pada View nomer 4b.

Query

```
DELETE FROM PROJECT_UNDER_10JUNE_V
WHERE job_name = 'HRD';
```

Screenshot

```
SQL> DELETE FROM PROJECT_UNDER_10JUNE_V
  2 WHERE job_name = 'HRD';
1 row deleted.

SQL> SELECT * FROM PROJECT_UNDER_10JUNE_V;
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      START_DATE END_DATE
----- ---------
 21 Software Engineer      Maya
 22 Bussiness Dev          Aga
 23 IT Support              Nara
 25 General Affair         Tara
 41 Admin                  Roni
 42 GA Draftter            Yunita
 43 IT Support              Rainy
  2 Front-End Developer    Sterling
  5 NOC                     Sterling

9 rows selected.

SQL>
```

Analisa :

Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **delete**. Data yang didelete disini adalah `job_name = 'HRD'` dengan mengacu kondisi pada klausula where pada view **PROJECT_UNDER_10JUNE_V**.



Laporan Sementara

Langkah Keempatbelas

Menapkan perintah DML pada View nomer 4c.

Query

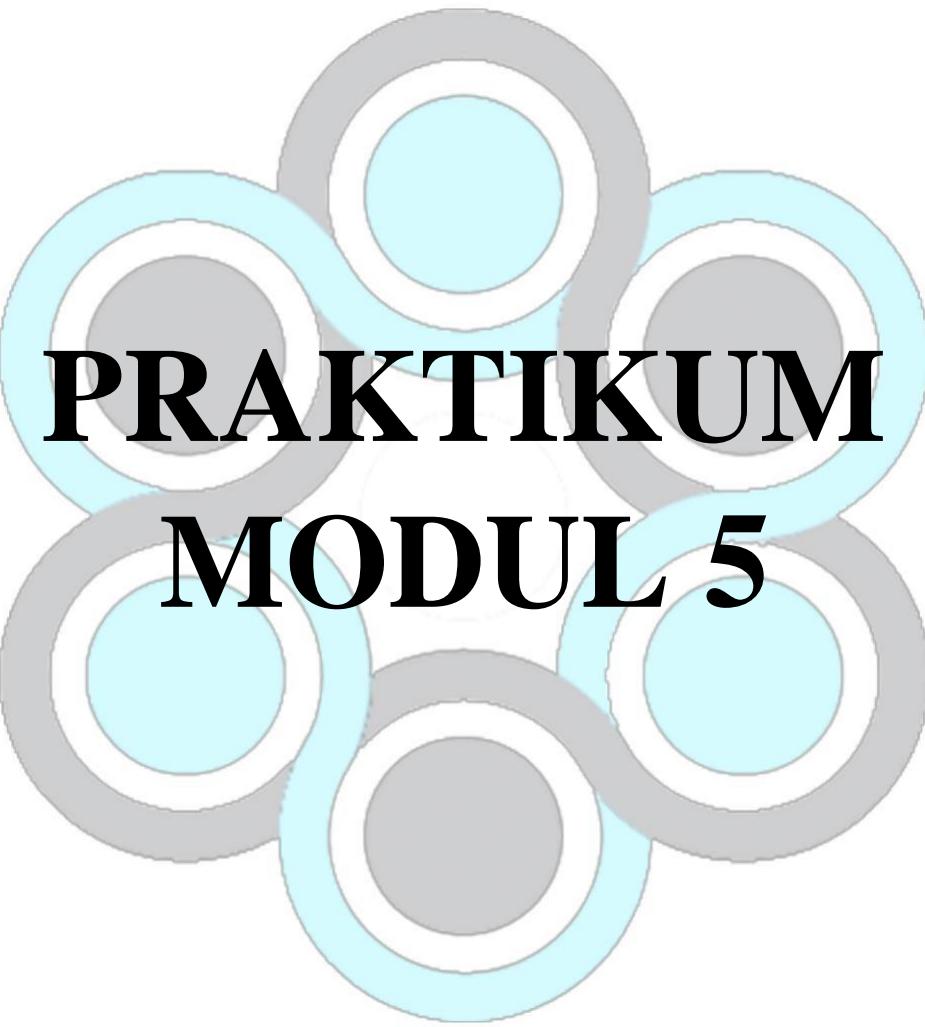
```
SELECT * FROM DUE_DATE_AFTER_30MAY ORDER BY  
id_project DESC;
```

Screenshot

```
SQL> SELECT * FROM DUE_DATE_AFTER_30MAY ORDER BY id_project DESC;  
ID_PROJECT JOB_NAME          EMPLOYEE_NAME      DESCRIPTION           EMAIL  
-----  
45 Admin HRD                Moza              Input data employee    moza@gmail.com  
44 General Affair           Yuna              Manage office tools  yuna@gmail.com  
43 IT Support                Rainy             Maintain network peripheral rainy@gmail.com  
42 GA Drafter               Yuna              Collect and submit tender yuna@gmail.com  
41 Admin IT                 Roni              Input ticket and generate ticket report roni@gmail.com  
SQL>
```

Analisa :

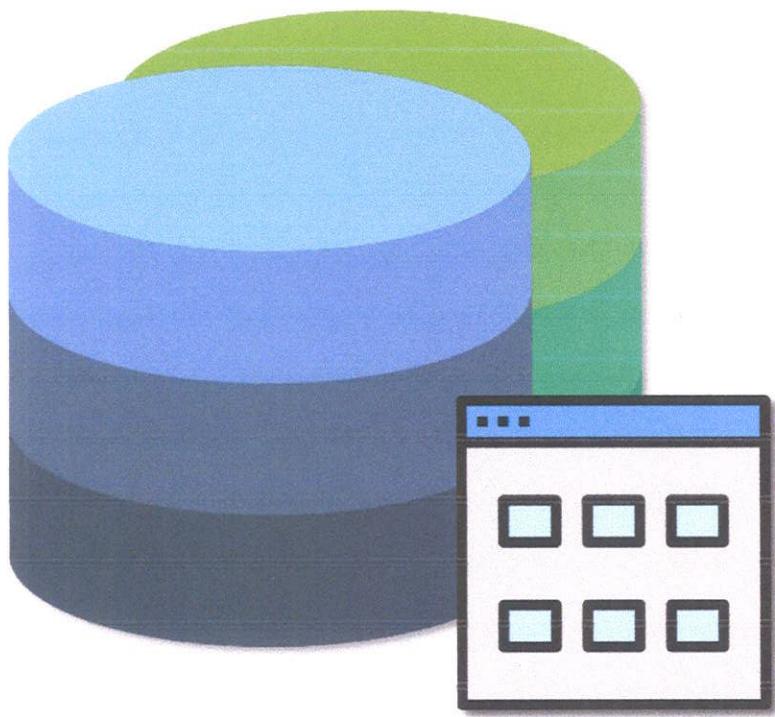
Query di atas digunakan untuk menjalankan salah satu perintah DML yaitu **select**. Data yang ditampilkan disini adalah seluruh data dari view yang sudah kita buat yakni **DUE_DATE_AFTER_30MY** dengan tambahan penerapan fungsi **ORDER BY DESC** pada **id_project**.



PRAKTIKUM MODUL 5

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020

LAPORAN PRAKTIKUM

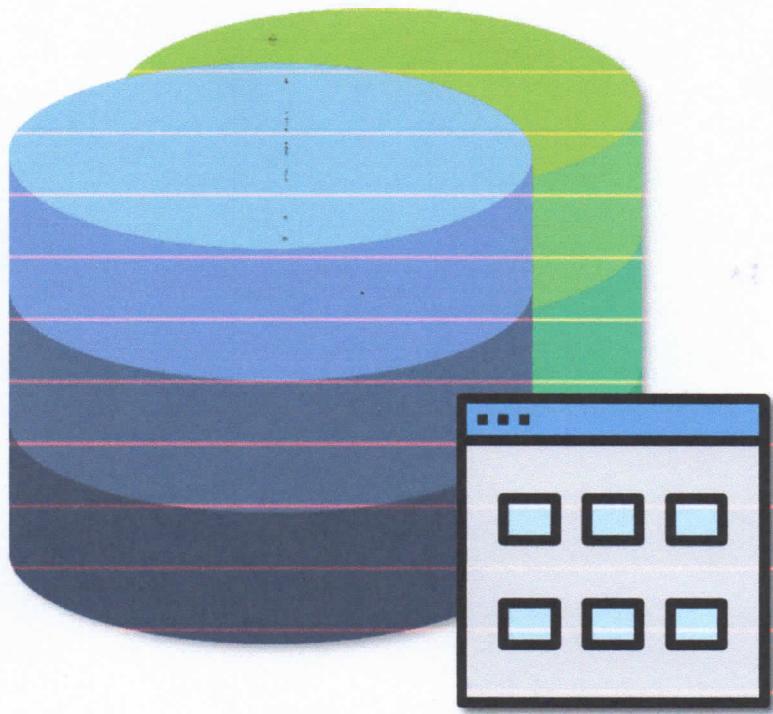


PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmad Muchlasin
NPM : 06.2018.1.06941
MODUL : V

--

LAPORAN PENDAHULUAN



PRAKTIKUM BASIS DATA PERIODE V - Tahun 2019/2020

NAMA : Achmad Muchlaem
NPM : 06.2018.1.06941
MODUL : V

--



Tugas Pertanyaan

Modul V

5.1 Tujuan

- Praktikan dapat mengetahui jarak Oracle Database dengan Java menggunakan IDE NetBeans.

5.2 Kriteria Penilaian

- Praktikan dapat mengimplementasikan Project dengan database.

5.3 Soal Pendahuluan

Pada bab ini kita akan mempelajari tentang cara mengkoneksikan program yang kita bangun dengan Database. Pada kasus ini kita akan membuat program dengan bahasa pemrograman Java dan Database yang digunakan adalah Oracle Database. Kasus yang kita gunakan adalah membuat aplikasi management transaksi dengan desain database yang sudah disediakan di Modul.

Sebelum membuat aplikasi kita harus mempersiapkan library / package yang kita gunakan untuk mengejarni koneksi Database Oracle yakni JDBC. Setelah kita menyiapkan data library tersebut, kita dapat langsung membuat projek baru di IDE NetBeans dan buat folder dengan nama "lib" yang berisi file library yang kita siapkan sebelumnya.

Untuk persiapan database pastikan sudah menyelesaikan langkah-langkah pada modul sebelumnya dan pastikan sudah ada Sequence ID-PEMBELIAN. Pada kasus ini kita akan melakukan hal yang sama seperti mencari topi dengan user baru yakni oracle java dengan password



Tugas Pertanyaan

Oracle Java 123. Struktur folder akan menjadi seperti berikut:

```
-- lib  
-- nl project  
-- src  
-- build.xml  
-- manifest.mf
```

Selanjutnya, silahkan buat Package dan Class sesuai struktur berikut :

```
-- controller  
--- - Transaksi.java  
-- database  
--- - Koneksi.java  
-- model  
--- - Detail_Pemesanan.java  
--- - Kategori.java  
--- - Kurir.java  
--- - Pegawai.java  
--- - Pelanggan.java  
--- - Pemasok.java  
--- - Pemesanan.java  
--- - Produk.java  
-- oraclejava  
--- - Oraclejava.java  
-- view  
--- - TransaksiView.java
```



Tugas Pertanyaan

Untuk transaksi view, java classnya adalah bentuk JFrame

Form dengan tampilan seperti berikut :

	Title 1	Title 2	Title 3	Title 4	Simpang
Belanggan	:				
Kegawai	:				
Kurir	:				
Alamat	:				

Tgl Pengiriman :

Produk :
Jumlah :
Diskon :
Total :

Total Hcnga

Title 1	Title 2	Title 3	Title 4

[View Detail](#)





Tugas Pertanyaan

Keterangan :

Komponen	Nama Variabel
ComboBox Pelanggan	cbPelanggan
ComboBox Pegawai	cbPegawai
ComboBox Kurir	cbKurir
TextField Alamat	tfAlamat
TextField Tgl Pengiriman	tfTglPengiriman
ComboBox Produk	cbProduk
TextField Jumlah	tfJumlah
TextField Diskon	tfDiskon
Button Tambah	btnTambah
Button Simpan	btnSimpan
Button Lihat Detail	btnDetail
TextField Total Harga	tfTotalHarga
Table Keranjang	tslKeranjang
Table Transaksi	tslTransaksi
TextArea Detail Transaksi	taDetailPesanan

Setelah itu kita siapkan logika programnya. Pertama kita akan menulis kode untuk koneksi dengan database di class `Koneksi.java` pada package database. Tulis kodennya seperti dibawah ini:

package database;

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Connection;
import java.sql.Connection;
import java.sql.Connection;
```



Tugas Pertanyaan

```
public class Koneksi {
    private Connection connect;
    private Statement db;
    private String database = "oraclejava";
}

public Koneksi() {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("Class driver ditemukan!");
        try {
            connect = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "oraclejava", "oraclejava123");
            System.out.println("Koneksi Database Sukses");
        } catch (SQLException se) {
            System.out.println("Koneksi Database Gagal: " + se);
        }
    } catch (ClassNotFoundException err) {
        System.out.println("Class driver tidak ditemukan, terjadi kesalahan pada: " + err);
    }
}

public ResultSet GetData(String sql) {
    try {
        db = connect.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
        return db.executeQuery(sql);
    } catch (SQLException e) {
        return null;
    }
}
```



Tugas Pertanyaan

```
public int ManipulasiData (String sql) {
    try {
        db = connect.createStatement();
        return db.executeUpdate(sql);
    } catch (SQLException e) {
        return 0;
    }
}
```

Keterangan :

Pada class ini kita mempunyai 3 fungsi yakni 'Koneksi', 'Get Data' dan 'manipulasi Data'. Koneksi sendiri berguna untuk membuat koneksi pada Database Oracle. Fungsi ini hanya kita jalankan 1 kali pada Objek. Fungsi 'GetData' adalah fungsi yang digunakan untuk mengambil data dari database, biasanya kita menggunakan query select untuk mengambil data dan menampungnya dalam sebuah objek ResultSet. Fungsi yang terakhir adalah 'Manipulasi Data' yang berguna untuk mengubah data di database. Biasanya kita menggunakan query insert, update dan delete.

Sesudah selesai menulis konvensi kita akan membuat class Data yang akan menampung data dari database untuk diolah di lingkungan program pada java. Maknukham kode berikut pada setiap kelasnya :

Pengembangan Java

```
package model;
```



Tugas Pertanyaan

```
import java.util.Date;
```

```
public class Karyawan {  
    private Integer id_Karyawan;  
    private String Nama_Depan;  
    private String Nama_Belakang;  
    private Date Tanggal_Lahir;  
    private String Alamat;  
    private Integer Kode_Pos;  
    private String No_Telp;
```

```
    public Integer getId_Karyawan() {  
        return id_Karyawan;
```

```
}
```

```
    public void setId_Karyawan(Integer id_Karyawan) {  
        this.id_Karyawan = id_Karyawan;
```

```
} /
```

```
    public String getNama_Depan() {  
        return Nama_Depan;
```

```
}
```

```
    public void setNama_Depan(String Nama_Depan) {  
        this.Nama_Depan = Nama_Depan;
```

```
}
```

```
    public String getNama_Belakang() {  
        return Nama_Belakang;
```

```
}
```



Tugas Pertanyaan

```
public void setNama_Kelakeng (String Nama_Belakeng) {  
    this.Nama_Kelakeng = Nama_Belakeng;  
}
```

```
public Note getTanggal_Lahir () {  
    return Tanggal_Lahir;  
}
```

```
public void setTanggal_Lahir ( Date Tanggal_Lahir ) {  
    this.Tanggal_Lahir = Tanggal_Lahir;  
}
```

```
public String getAlamat () {  
    return Alamat;  
}
```

```
public void setAlamat ( String Alamat ) {  
    this.Amat = Alamat;  
}
```

```
public Integer getKode_Pos () {  
    return Kode_Pos;  
}
```

```
public void setKode_Pos ( Integer Kode_Pos ) {  
    this.Kode_Pos = Kode_Pos;  
}
```

```
public String getNo_Telp () {  
    return No_telp;  
}
```



Tugas Pertanyaan

```
public void setNo-Telp (String No-Telp) {  
    this.No-Telp = No-Telp;  
}
```

Pelanggan.java

```
package model;
```

```
import java.util.Date;
```

```
public class Pelanggan {  
    private Integer Id-Pelanggan;  
    private String Nama-Repar;  
    private String Nama-Blokang;  
    private Date Tanggal-Lahir;  
    private String Alamat;  
    private Integer Kode-Pes;  
    private String No-Telp;
```

```
    public Integer getId-Pelanggan () {  
        return Id-Pelanggan;  
    }
```

```
    public void setId-Pelanggan (Integer Id-Pelanggan) {  
        this.Id-Pelanggan = Id-Pelanggan;  
    }
```

```
    public String getNama_Repar () {  
        return Nama-Repar;  
    }
```



Tugas Pertanyaan

```
public void setNama - Nipan ( String Nama - Nipan ) {
```

```
    this . Nama - Nipan = Nama - Nipan ;
```

```
}
```

```
public String getNama - Belakang () {
```

```
    return Nama - Belakang ;
```

```
}
```

```
public void setNama - Belakang ( String Nama - Belakang ) {
```

```
    this . Nama - Belakang = Nama - Belakang ;
```

```
}
```

```
public Date getTanggal - Lahir () {
```

```
    return Tanggal - Lahir ;
```

```
}
```

```
public void setTanggal - lahir ( Date Tanggal - Lahir ) {
```

```
    this . Tanggal - lahir = Tanggal - lahir ;
```

```
}
```

```
public String getAlamat () {
```

```
    return Alamat ;
```

```
}
```

```
public void setAlamat ( String Alamat ) {
```

```
    this . Alamat = Alamat ;
```

```
}
```

```
public Integer getKode - Pos () {
```

```
    return Kode - Pos ;
```

```
{
```



Tugas Pertanyaan

```
public String getNo_Telp () {  
    return No_Telp;  
}
```

```
public void setNo_Telp (String No_Telp) {  
    this.No_Telp = No_Telp;  
}
```

```
}
```

```
Kurir.java
```

```
package model;
```

```
public class Kurir {  
    private Integer Id_Kurir;  
    private String Alamat_Perserahaan;  
    private String No_Telp;
```

```
    public Integer getId_Kurir () {  
        return Id_Kurir;  
    }
```

```
    public void setId_Kurir (Integer id_Kurir) {  
        this.Id_Kurir = id_Kurir;  
    }
```

```
    public String getAlamat_Perserahaan () {  
        return Alamat_Perserahaan;  
    }
```



Tugas Pertanyaan

```
public void setNama_Persahaan (String Nama_Persahaan) {
```

```
    this.Nama_Persahaan = Nama_Persahaan;
```

```
}
```

```
public String getNo_Telp () {
```

```
    return No_Telp;
```

```
}
```

```
public void setNo_Telp (String No_Telp) {
```

```
    this.No_Telp = No_Telp;
```

```
?
```

```
}
```

Kategori.java

```
package model;
```

```
public class Kategori {
```

```
    private Integer Id_Kategori;
```

```
    private String Nama_Kategori;
```

```
    public Integer getId_Kategori () {
```

```
        return Id_Kategori;
```

```
    }
```

```
    public void setId_Kategori (Integer Id_Kategori) {
```

```
        this.Id_Kategori = Id_Kategori;
```

```
    }
```



Tugas Pertanyaan

```
public String getNamaKategori() {
    return NamaKategori;
}

public void setNamaKategori(String NamaKategori) {
    this.NamaKategori = NamaKategori;
}
```

Remasch.java

```
package model;
```

```
public class Remasch {
    private Integer Id_Remasch;
    private String Nama_Pensahaan;
    private String Alamat;
    private Integer Kode_Pos;
    private String No_Telp;
```

```
public Integer getId_Remasch() {
    return Id_Remasch;
}
```

```
public void setId_Remasch(Integer Id_Remasch) {
    this.Id_Remasch = Id_Remasch;
}
```

```
public String getNama_Pensahaan() {
    return Nama_Pensahaan;
}
```



Tugas Pertanyaan

```
public void setNama_Perschaen (String Nama_Perschaen) {  
    this.Nama_Perschaen = Nama_Perschaen;  
}
```

```
public String getAlamat() {  
    return Alamat;  
}
```

```
public void setAlamat (String Alamat) {  
    this.Alatat = Alamat;  
}
```

```
public Integer getKode_Pos () {  
    return Kode_Pos;  
}
```

```
public void setKode_Pos (Integer Kode_Pos) {  
    this.Kode_Pos = Kode_Pos;  
}
```

```
public String getNo_Telp () {  
    return No_Telp;  
}
```

```
public void setNo_Telp (String No_Telp) {  
    this.No_Telp = No_Telp;  
}
```



Tugas Pertanyaan

Renesenem . java

package model;

import java.util.ArrayList;

import java.util.Date;

public class Renesenem {

private Insyer ld_Renesenem;

private Pelanggan pelanggan;

private Rejawi pejawi;

private Kurir kurir;

private Date Tanggal_Renesenem;

private Date Tanggal_Renjinew;

private String Alamat_Renjinew;

private Date Harga_Toko;

private ArrayList< basii_Renesenem > arahBasiil_Renesenem;

public Insyer getld_Renesenem () {

return ld_Renesenem;

}

public void setld_Renesenem (Insyer ld_Renesenem) {

this. ld_Renesenem = ld_Renesenem;

}

public Pelanggan getPelanggan () {

return pelanggan;

}



Tugas Pertanyaan

public void setPelanggan (Pelanggan pelanggan) {

this.pelanggan = pelanggan;

}

public Pegawai getPegawai() {

return pegawai;

}

public void setPegawai (Pegawai pegawai) {

this.pegawai = pegawai;

}

public Kurir getKurir() {

return kurir;

}

public void setKurir (Kurir kurir) {

this.kurir = kurir;

}

public Date getTanggal_Pemesanan () {

return Tanggal_Pemesanan;

}

public void setTanggal_Pemesanan (Date Tanggal_Pemesanan) {

this.Tanggal_Pemesanan = Tanggal_Pemesanan;

}

public Date getTanggal_Pengiriman () {

return Tanggal_Pengiriman;

}



Tugas Pertanyaan

```
public void setTanggal - Pengiriman (Date Tanggal - Pengiriman) {
```

```
this.Tanggal - Pengiriman = Tanggal - Pengiriman;
```

```
}
```

```
public String getAlamat - Pengiriman () {
```

```
return Alamat - Pengiriman;
```

```
}
```

```
public void setAlamat - Pengiriman (String Alamat - Pengiriman) {
```

```
this.Altamet - Pengiriman = Alamat - Pengiriman;
```

```
}
```

```
public Double getHarga - Total () {
```

```
return Harga - Total;
```

```
}
```

```
public void setHarga - Total (Double Harga - Total) {
```

```
this.Harga - Total = Harga - Total;
```

```
}
```

```
public void setArrDetail - Pemesanan (ArrayList<Detail - Pene
```

```
seman> arrDetail - Pemesanan) {
```

```
this.arrDetail - Pemesanan = arrDetail - Pemesanan;
```

```
}
```

```
public ArrayList<Detail - Pemesanan> getArrDetail -
```

```
Pemesanan () {
```

```
return arrDetail - Pemesanan;
```

```
}
```

```
3
```



Tugas Pertanyaan

Detailed_Remesanan.java

```
public class Detailed_Remesanan {
```

```
    private Produk produk;
```

```
    private Integer jumlah;
```

```
    private Double diskon;
```

```
    public Produk getProduk () {
```

```
        return produk;
```

```
}
```

```
    public void setProduk (Produk produk) {
```

```
        this.produk = produk;
```

```
}
```

```
    public Integer getJumlah () {
```

```
        return jumlah;
```

```
}
```

```
    public void setJumlah (Integer jumlah) {
```

```
        this.jumlah = jumlah;
```

```
}
```

```
    public Double getDiskon () {
```

```
        return diskon;
```

```
}
```

```
    public void setDiskon (Double diskon) {
```

```
        this.diskon = diskon;
```

```
}
```

```
}
```



Tugas Pertanyaan

Silahkan membuat class Metode kita akan menulis pada class Transaksi.java pada package Controller untuk memproses logika program. Silahkan tulis kode berikut :

```
package controller;
```

```
import model.Retail_Kemasan;
import model.Kategori;
import model.Kurir;
import model.Pegawai;
import model.Pelanggan;
import model.Pembach;
import model.Ruang;
import model.Produk;
import database.Koneksi;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
```

```
public class Transaksi {
    Koneksi koneksi;
    ArrayList<Pelanggan> arrPelanggan;
    ArrayList<Pegawai> arrPegawai;
    ArrayList<Kurir> arrKurir;
    ArrayList<Produk> arrProduk;
    ArrayList<Ruang> arrRuang;
```



Tugas Pertanyaan

```
public Transaksi() throws SQLException {
    this.keluar = new Keluar();
    this.curPelanggan = new Pelanggan();
    this.curPegawai = new Pegawai();
    this.curKeranjang = new Keranjang();
    this.curPembelian = new Pembelian();
}
```

```
public ArrayList<Produk> getDataProduk() throws
SQLException {
    this.curProduk.clear();
    ResultSet rs = this.koneksi.createStatement().executeQuery("SELECT * FROM
PRODUK JOIN KATEGORI ON PRODUK.ID_KATEGORI =
KATEGORI.ID_KATEGORI JOIN PEMASOK ON PRODUK.
ID_PEMASOK = PEMASOK.ID_PEMASOK");
    while (rs.next()) {
        Pemasok pemasok = new Pemasok();
        pemasok.setIdPemasok(rs.getInt("ID_PEMASOK"));
        pemasok.setNamaPemasok(rs.getString("NAMA_"
+ "PEMASOK"));
        pemasok.setAlamat(rs.getString("ALAMAT"));
        pemasok.setIdKodePos(rs.getInt("KODE_POS"));
        pemasok.setNoTelp(rs.getString("NO_TELP"));
    }
}
```

```
Kategori kategori = new Kategori();
kategori.setIdKategori(rs.getInt("ID_KATEGORI"));
kategori.setNamaKategori(rs.getString("NAMA_"
+ "KATEGORI"));
```



Tugas Pertanyaan

```
Produk produk = new Produk();
produk.setId_Produk(rs.getInt("ID_Produk"));
produk.setNamaProduk(rs.getString("NAMA_Produk"));
produk.setKategori(rs.getString("KATEGORI"));
produk.setNama_Produk(rs.getString("NAMA_Produk"));
produk.setHargaSatuan(rs.getInt("HARGA_SATUAN"));
produk.setStock_Produk(rs.getInt("STOCK_Produk"));
this.arrProduk.add(produk);
}

return this.arrProduk;
}
```

```
public ArrayList<Pelanggan> getNamaPelanggan() throws
SQLException {
    this.arrPelanggan.clear();
    ResultSet rs = this.koneksi.createStatement()
        .executeQuery("SELECT * FROM
        PELANGGAN");
    while(rs.next()) {
        Pelanggan pelanggan = new Pelanggan();
        pelanggan.setId_Pelanggan(rs.getInt("ID_PELANGGAN"));
        pelanggan.setNama_IstSpm(rs.getString("NAMA_NEPALI"));
        pelanggan.setNama_Belakang(rs.getString("NAMA_BELAKANG"));
        pelanggan.setTanggallahir(rs.getDate("TANGGAL_LAHIR"));
        pelanggan.setAlamat(rs.getString("ALAMAT"));
        pelanggan.setKode_Pos(rs.getInt("KODE_POS"));
        pelanggan.setNo_Telp(rs.getString("NO_TELP"));
        this.arrPelanggan.add(pelanggan);
    }
    return this.arrPelanggan;
}
```



Tugas Pertanyaan

```
public ArrayList< Pegawai > getBata Pegawai() throws SQLException
```

```
{
```

```
    this . arr Pegawai . clear () ;
```

```
    Result Set rs = this . koneksi . GetMateri (" SELECT * FROM PEGAWAI ");
```

```
    while ( rs . next ()) {
```

```
        Pegawai pegawai = new Pegawai ();
```

```
        pegawai . setId_Pegawai ( rs . getInt (" ID_PEGAWAI "));
```

```
        pegawai . setNama_Altama - Pegawai ( rs . getString (" NAMA_ALTAMA "));
```

```
        pegawai . setNama_Belakang ( rs . getString (" NAMA_BELAKANG "));
```

```
        pegawai . setTempatLahir ( rs . getAttribute (" TEMPAT_LAHIR "));
```

```
        pegawai . setAlamat ( rs . getString (" ALAMAT "));
```

```
        pegawai . setKode_Ras ( rs . getInt (" KODE_RAS "));
```

```
        pegawai . setNo_Telp ( rs . getString (" NO_TELP "));
```

```
        this . arr Pegawai . add ( pegawai );
```

```
}
```

```
    return this . arr Pegawai ;
```

```
}
```

```
public ArrayList< Kurir > artKurir () throws SQLException {
```

```
    this . arrKurir . clear ();
```

```
    Result Set rs = this . koneksi . GetMateri (" SELECT * FROM KURIR ");
```

```
    while ( rs . next ()) {
```

```
        Kurir kurir = new Kurir ();
```

```
        kurir . setId_Kurir ( rs . getInt (" ID_KURIR "));
```

```
        kurir . setNama_Kerjanya ( rs . getString (" NAMA_KERJAYA ));
```

```
        kurir . setNo_Telp ( rs . getString (" NO_TELP "));
```

```
        this . arrKurir . add ( kurir );
```

```
}
```

```
    return this . arrKurir ;
```

```
}
```



Tugas Pertanyaan

```
public ArrayList<Pemesanan> getDataPemesanan() throws  
SQLException {  
    this.arrPemesanan.clear();  
  
    Result Set rs = this.koneksi.QUERY("SELECT ID_PELANGGAN, ID_  
PELANGGAN, NAMA_DEPAN, AS NAMA_DEPAN_  
PELANGGAN, NAMA_BELAKANG AS NAMA_BELAKANG_  
PELANGGAN, PELANGGAN.TANGGAL_LAHIR AS TANGGAL_LAHIR_  
PELANGGAN, PELANGGAN.ALAMAT AS ALAMAT_PELANGGAN,  
PELANGGAN.KODE_POS AS KODE_POS_PELANGGAN,  
PELANGGAN.NO_TELP AS NO_TELP_PELANGGAN, RECAWAID  
ID_PEGAWAI, PEGAWAI.NAMA_DEPAN AS NAMA_DEPAN_PEGAWAI,  
pegawai.NAMA_BELAKANG AS NAMA_BELAKANG_PEGAWAI,  
pegawai.TANGGAL_LAHIR AS TANGGAL_LAHIR_PEGAWAI,  
PEGAWAI.ALAMAT AS ALAMAT_PEGAWAI, PEGAWAI.KODE_POS  
AS KODE_POS_PEGAWAI, PEGAWAI.NO_TELP AS NO_TELP_PEGAWAI,  
PESANAN.*, KURIR.* FROM PESANAN JOIN PELANGGAN ON  
PESANAN.ID_PELANGGAN = PELANGGAN.ID_PELANGGAN  
JOIN KURIR ON PESANAN.ID_KURIR = KURIR.ID_KURIR  
JOIN PEGAWAI ON PESANAN.ID_PEGAWAI = PEGAWAI.  
ID_PEGAWAI ORDER BY ID_PESANAN DESC");  
  
    while (rs.next()) {  
        Pelanggan pelanggan = new Pelanggan();  
        pelanggan.setIdPelanggan(rs.getInt("ID_PELANGGAN"));  
        pelanggan.setNamaDepan(rs.getString("NAMA_DEPAN"));  
        pelanggan.setNamaBelakang(rs.getString("NAMA_BELAKANG_PELANGGAN"));  
  
        pelanggan.setTanggalLahir(rs.getDate("TANGGAL_LAHIR_PELANGGAN"));  
        pelanggan.setAlamat(rs.getString("ALAMAT"));  
        pelanggan.setKodePos(rs.getInt("KODE_POS_PELANGGAN"));  
    }  
}
```



Tugas Pertanyaan

pelanggan. setNo_Telp (rs.getString("NO_TELP_PELANGGAN"));

Pegawai pegawai = new Pegawai();

pegawai. setId_Pegawai (rs.getInt("ID_PEGAWAI"));

pegawai. setNama_Depan (rs.getString("NAMA_DEPAN_PEGAWAI"));

pegawai. setTanggal_Lahir (rs.getDate("TANGGAL_LAHIR_PEGAWAI"));

pegawai. setAlamat (rs.getString("ALAMAT_PEGAWAI"));

pegawai. setKode_Pos (rs.getInt("KODE_POS_PEGAWAI"));

pegawai. setNo_Telp (rs.getString("NO_TELP_PEGAWAI"));

Kurir kurir = new Kurir();

kurir. setId_Kurir (rs.getInt("ID_KURIR"));

kurir. setNama_Rawabahan (rs.getString("NAMA_PENGUSAHAAN"));

kurir. setNo_Telp (rs.getString("NO_TELP"));

Pemesanan pemesanan = new Pemesanan();

pemesanan. setId_Kemesanan (rs.getInt("ID_PEMESANAN"));

pemesanan. setPelanggan (pelanggan);

pemesanan. setPegawai (pegawai);

pemesanan. setKurir (kurir);

pemesanan. setTanggal_Pemesanan (rs.getDate("TANGGAL_PESANAN"));

pemesanan. setTanggal_Renjiriman (rs.getDate("TANGGAL_PENGIRIMAN"));

pemesanan. setAlamat_Pengirim (rs.getString("ALAMAT_PENGIRIMAN"));



Tugas Pertanyaan

```
pemesanan.setHarga_Total(rs.getDouble("HARGA_TOTAL"));
```

```
ResultSet rsDetail_Pemesanan = this.koneksi.getQuery("SELECT *  
FROM DETAIL_PEMESANAN JOIN produk ON DETAIL_PEMESANAN.  
ID_PRODUK = PRODUK.ID_PRODUK JOIN PEMASOK ON  
PRODUK.ID_PEMASOK = PEMASOK.ID_PEMASOK JOIN KATEGORI  
ON KATEGORI.ID_KATEGORI = KATEGORI.ID_KATEGORI WHERE  
DETAL_PEMESANAN.ID_PEMESANAN = " + n.getString  
("ID_PEMESANAN"));
```

```
ArrayList<Detail_Pemesanan> dp = new ArrayList();
```

```
while(rsDetail_Pemesanan.next()) {  
    Remasuk pemesok = new Remasuk();  
    pemesok.setId_Remasuk(rsDetail_Pemesanan.getInt  
("ID_PEMASOK"));  
    pemesok.setNama_Remasuk(rsDetail_Pemesanan.  
getString("NAMA_PEMASOK"));  
    pemesok.setAlamat(rsDetail_Pemesanan.getString  
("ALAMAT"));  
    pemesok.setKode_Pos(rsDetail_Pemesanan.getInt  
("KODE_POS"));  
    pemesok.setNo_Telp(rsDetail_Pemesanan.getString  
("NO_TELP"));
```

```
Kategori kategori = new Kategori();
```

```
kategori.setId_Kategori(rsDetail_Pemesanan.getInt  
("ID_KATEGORI"));  
kategori.setNama_Kategori(rsDetail_Pemesanan.getString  
("NAMA_KATEGORI"));
```



Tugas Pertanyaan

```
Produk produk = new Produk();
produk.setId_Produk(rsDetail_Pemesanan.getInt("ID_Produk"));
produk.setNama_produk(pemesanan.getString("NAMA_Produk"));
produk.setKategori(rsDetail_Pemesanan.getString("KATEGORI"));
produk.setNama_Produk(rsDetail_Pemesanan.getString("NAMA_Produk"));
produk.setHarga_Satuan(rsDetail_Pemesanan.getInt("HARGA_SATUAN"));
produk.setStock_Produk(rsDetail_Pemesanan.getInt("STOK_Produk"));

Detal_Pemesanan detail_pemesanan = new Detal_Pemesanan();
detail_pemesanan.setProduk(produk);
detail_pemesanan.setJumlah(rsDetail_Pemesanan.getInt("JUMLAH"));
detail_pemesanan.setDiskon(rsDetail_Pemesanan.getDouble("DISKON"));
dp.add(detail_pemesanan);
}

pemesanan.setArr_Detal_Pemesanan(dp);
this.arr_Pemesanan.add(pemesanan);
}

return this.arr_Pemesanan;
}

public void insertTransaksi(Pemesanan pemesanan) {
try {
String datePemesanan = new SimpleDateFormat("dd/mm/yyyy").format(pemesanan.getTanggal_Pemesanan());
String intgrtRujukan = new SimpleDateFormat("dd/mm/yyyy").format(pemesanan.getTanggal_Rujukan());
}
```



Tugas Pertanyaan

```
this.koneksi. ManipulasiData ("INSERT INTO PEMESANAN  
VALUES (?) IN_PEMESANAN, NEXTVAL " + pemesanan.getPelanggan (),  
getId_Pelanggan () + ", " + pemesanan.getKurir (),  
getId_Kurir () + ", TO_DATE (" + date_pemesanan + "dd/mm/yyyy/  
yyyy"), TO_DATE (" + date_kiriman + "dd/mm/yyyy),  
" + pemesanan.getAlamat_Kiriman () + ", " +  
pemesanan.getHarga_Tot, (, to_string () + ");  
this.koneksi . ,
```

```
Result Set rs = this.koneksi. GetData ("SELECT ID_PEMESANAN,  
CURVAL FROM DUAL");  
rs.next();  
int id_pemesanan = rs.getInt ("CURVAL");  
for (Detail_Pemesanan D Pemesanan.getArrDetail () -  
pemesanan ()) {  
this.koneksi. ManipulasiData ("INSERT INTO DETAIL_  
PEMESANAN VALUES (" + p. getProduk () . getId_Produk ()  
+ ", " + id_pemesanan + ", " + p.getJumlah () + ", " +  
p.getHarga () + ")");  
this.koneksi. ManipulasiData ("UPDATE Produk  
SET stock_produk = stock_produk - " + p.jumlah ()  
WHERE id_produk = " + p.getProduk () . getId_Produk ());
}  
} catch (SQLException ex) {  
System.out.println (ex);  
}  
}
```



Tugas Pertanyaan

Sekarang selesainya, maka kita akan mengambil tampilan yang sudah kita buat pada class TransaksiView.java dijelaskan program pada Transaksi.java

Pada class TransaksiView.java bukti obyek sesegera berikut :

```
private Transaksi transaksi;
private ArrayList<DetailPemesanan> arrKeranjang;
private Double hargaTotal;
```

Pada constructor tambahkan kode berikut :

```
this.hargaTotal = 0.0;
this.transaksi = new Transaksi();
this.arrKeranjang = new ArrayList<>();
this.showTablePemesanan();
this.showCaraBayarPelanggan();
this.showCaraBayarPegawai();
this.showCaraBayarKurir();
this.showCaraBayarProduk();
this.showTableKeranjang();
this.showHargaTotal();
```

Buktifungsinya sesegera berikut :

```
public void showTableKeranjang() {
    DefaultTableModel dtmKeranjang = new DefaultTableModel(
        new String[] {"Nama Produk", "Harga Satuan", "Jumlah", "Harga"}, 
        0);
    dtmKeranjang.setRowCount(0);
```



Tugas Pertanyaan

```
for (Detail_Pembelian dp : this.arrKeranjang) {
    dtmKeranjang.addRow (new String[] { dp.getProduk(),
        getNama() + Produk(), dp.getProduk(), dp.HargaSatuan(),
        toString(), dp.getJumlah(), toString(), dp.getDiskon(),
        toString() });
}
```

```
this.tbKeranjang.setModel (dtmKeranjang);
}
```

```
public void showCartBox(Pelanggan) throws SQLException {
    DefaultComboBoxModel dtmPelanggan = new Default
    ComboBoxModel();
}
```

```
for (Pelanggan p : this.transaksi.getDataPelanggan()) {
    dtmPelanggan.addElement (p.getNama_Kepem() +
        " " + p.getNama_Belakang());
}
```

```
this.cb_Pelanggan.setModel (dtmPelanggan);
}
```

```
public void showComboBox_Pegawai() throws SQLException {
    DefaultComboBoxModel model = new Default
    ComboBoxModel();
}
```

```
for (Rejawan p : this.transaksi.getDataPegawai()) {
    dtmPegawai.addElement (p.getNama_Bapak() +
        " " + p.getNama_Belakang());
}
```



Tugas Pertanyaan

this. cbRejawar. setModel (cbdmRejawari),

}

public void showCarboBoxKurir() throws SQLException {

DefaultCarboBoxModel cbmKurir = new DefaultCarboBoxModel();

for (Kurir k : this.transaksi.getKurir()) {

cbmKurir.addElement(k.getNama_Penschaaw());

}

this. cbKurir.setModel (cbmKurir);

}

public void ShowCarboBoxProduk() throws SQLException {

DefaultCarboBoxModel cbmProduk = new DefaultCarboBoxModel();

for (Produk p : this.transaksi.getNama_Produk()) {

cbmProduk.addElement(p.getNama_Produk() + "

" + p.getStock_Produk() + ")");

}

this. cbProduk.setModel (cbmProduk);

}

public void ShowHargaTotal() {

this. lfTotal_Harga. setText (this. hrgatotal.toRupiah());

}



Tugas Pertanyaan

Pada tampilan bkn simpel untuk bukaan action parameter sebagai berikut :

```
private void btnSimpanActionPerformed(ActionEvent e) {
```

```
    Pemesanan pesanan = new Pemesanan();  
    try {
```

```
        pesanan.setPelanggan((this.getTransaction().getNotaPelanggan()).  
        get(0).getPelanggan().getSelectedIndex());
```

```
        pesanan.setRejawali((this.getTransaction().getNotaRejawali()),  
        get(0).getRejawali().getSelectedIndex());
```

```
        pesanan.setKurir((this.getTransaction().getNotaKurir())  
        .get(0).getKurir().getSelectedIndex());
```

```
        pesanan.setTanggol_Pengiriman(new Date());
```

```
        pesanan.setTanggal_Pengiriman(new SimpleDateFormat  
        ("dd/mm/yyyy").parse(this.tanggalPengiriman.getTxt()));
```

```
        pesanan.setAlamat_Pengiriman(this.tfAlamat.get  
        text());
```

```
        pesanan.setHarga_Total(this.hargaTotal);
```

```
        pesanan.setStatus_Batal_Pemesanan((this.aktarangka));
```

```
        ((this.getTransaction().getTransaksi(pesanan)));
```

```
        this.addRowTablePemesanan();
```

```
        this.arra_Kategori = new ArrayList();
```



Tugas Pertanyaan

```
this.hargatotal = 0;0;
```

```
this.showHargaTotal();
```

```
this.showBelanjaProduk();
```

```
{ catch (SQLException | ParseException e) { }
```

```
System.out.println(e);
```

```
}
```

```
}
```

Pada btn tambah temambahkan kode berikut :

```
private void bentambahActionPerformed (java.awt.event.ActionEvent evt)
```

```
{
```

```
try {
```

```
BelanjaPembelian dp = new BelanjaPembelian ();
```

```
dp.setJumlah (Integer.parseInt (tfJumlah.getText ()));
```

```
dp.setProduk (this.transaksi.getDaftarProduk ().get  
(cbProduk.getSelectedIndex ()) .getHargaSatuan ()
```

```
);
```

```
if (this.tfBiliken.getText () .equals ("0")) {
```

```
dp.setDiskon (Double.parseDouble ("0"));
```

```
this.hargatotal += this.transaksi.getDaftarProduk (),
```

```
get (cbProduk.getSelectedIndex ()) .getHargaSatuan ()
```

```
);
```

```
} else {
```

```
Double diskon = Double.parseDouble (this.tfBiliken.  
getText ());

```

```
Integer hargatotalProduk = this.transaksi.
```

```
getDaftarProduk () .get (cbProduk.getSelectedIndex ()
```

```
) .getHargaSatuan () * Integer.parseInt
```

```
(tfJumlah.getText ());

```



Tugas Pertanyaan

Pausle harga Total Diskon = harga Total Produk + diskon;

elp. setDirn (dirn);

this.hargaTotal += hargaTotalProduk - hargaTotalDiskon;

3

this, Show Many Total () ;

this.unkeranjang.add(dlp));

this.showTableKerangjaring());

this. $\text{tfJumlah} \cdot \text{stText}(i, n)$;

this technique. set text (" ");

```
    } catch (SQLException ex) {
```

Legger, getLegger (TvenskaView.class.getName()), log(

```
!-arel SEVENT, null, ex);
```

3

1

Paru von Nekar temschken mole berukt.

```
private void SetDetail(ActionPerformedEvent evt) {
```

2

try {

Persemen persemen = this.transaksi.getDaftarPersemen();

```
get (this.tbl_Pesanan.get SelectedRow ()) ;
```

String takes = " " + " == == == REVERSED == == " +

"(n)" + "Mama Repen": " + person + get Pelargonium). get them Repen ()

+ "W" + "Name Belohnung": "+ plausiblen. getBelohnung(). getName_Belohnung

+ "n" + "Terugval Lechir": " + newSimpleDateFormat("dd/MM/yyyy") format

(penerver getReferenzen(), getTanzd_Lichts()) + " u"

+ "Alonet : " + `newgen.setAlonet()`. getAlonet() + " \n "

+ "Kode Reg." + "t_nasional" mit Pfeil von 17. sit Ked-Reg(?) + "Pn"



Tugas Pertanyaan

- + "No. Telp": "+ pemesanan.getRekening().getNo_Telp() + "\n"
- + "==== PEGAWAI =====": "+ pemesanan.getRekening().getPegawai().getNama_Pegawai() + "\n"
- + "Nama Pegawai": "+ pemesanan.getRekening().getPegawai().getNama_Pegawai() + "\n"
- + "Nama Belanja": "+ pemesanan.getRekening().getPegawai().getNama_Belanja() + "\n"
- + "Tanggih Lahir": "+ pemesanan.getRekening().getTanggihLahir() + "\n"
- + "Alamat": "+ pemesanan.getRekening().getAlamat() + "\n"
- + "Kode Pos": "+ pemesanan.getRekening().getKode_Pos() + "\n"
- + "No.Telp": "+ pemesanan.getRekening().getNo_Telp() + "\n"
- + "==== KURIR =====": "+ pemesanan.getKurir().getNama_Kurir() + "\n"
- + "Nama Pengiriman": "+ pemesanan.getKurir().getNama_Kurir() + "\n"
- + "No.Telp": "+ pemesanan.getKurir().getKode_Telp() + "\n"
- + "==== PEMERINTAHAN =====": "+ pemesanan.getKurir().getNama_Kurir() + "\n"
- + "Tunggal Pemesanan": "+ new SimpleDateFormat("dd/MM/yyyy").format(pemesanan.getTunggal_Pengiriman()) + "\n"
- + "Alamat Pengiriman": "+ pemesanan.getAlamat_Pengiriman() + "\n"
- + "Harga total": "+ pemesanan.getHarga_Total() + "\n"
- + "Prodisk": "+ "\n";

```
ArrayList<DetailReiseantrag> dp = fahrsachen.getArrayListDetailReiseantrag();
Reiseantrag();
```

```

for (int i = 0; i < dp.size(); i++) {
    teks += "ln[" + (i+1) + ". " + "Pembelahan Bacsozi : " +
dp.get(i).getProduct().getPembelahan() .getKoma - proseskan() +
"ln"
    + "/ dan Pembelahan : " + dp.get(i).getProduct().getPembelahan().get -
Koma() + "ln"
    + "KodeBac Bacsozi " + dp.get(i).getProduct().getPembelahan().get -
KodeBac() + "ln"
    + "ln, total pembelahan : " + dp.get(i).getProduct().getPembelahan()

```



Tugas Pertanyaan

```
- get Mo_Pelip() + "\n"
+ "Mama Produk: " + dp.get(0).getNama_Produkt() + "\n"
+ "Harga Produk: " + dp.get(0).getHarga_Satuan() + "\n"
+ "Satu Produk: " + dp.get(0).getStok_Produkt() + "\n"
+ "Jumlah Beli: " + dp.get(0).getJumlah() + "\n"
+ "Bulan: " + dp.get(0).getBulan() + "\n"
```

}

```
this.jcDetailReservasi.setText(telp);
```

```
} catch (SQLException ex) {
```

```
System.out.println(ex);
```

}

}

Setelah semuanya selesai, tinggal kita panggil view tersebut bedakan
class mainnya yaitu OracleJava.java pada package oracleJava

```
package oracleJava;
```

```
import java.sql.SQLException;
```

```
import java.awt.AWTException;
```

```
import javax.swing.JFrame;
```

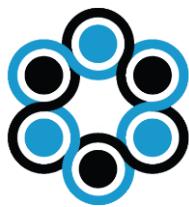
```
public class OracleJava {
```

```
public static void main (String[] args) throws  
SQLException, AWTException {
```

```
new MainView().show();
```

}

4



Tugas Praktikum

Soal Praktikum

1. Buat koneksi database Oracle dengan Java/Web berdasarkan project masing-masing.
2. Buatlah CRUD pada salah satu tabel (boleh menggunakan GUI, Console atau Web).
3. Terapkan salah satu konsep JOIN dan tampilkan pada project Anda.
4. Terapkan salah satu konsep VIEW dan tampilkan pada project Anda.

Ketentuan :

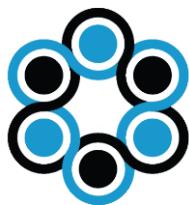
1. Gunakan SQLPlus pada terminal.
2. Kerjakan dalam waktu 150 Menit.

Petunjuk :

1. Bentuk penulisan laporan sementara sesuai template yang ada.
2. Source code dan soal diketik sesuai format yang ada.
3. Upload laporan sementara di classroom dengan batas waktu yang telah ditentukan.
4. Format penamaan upload file :

NPM_NamaPraktikan_LaporanSementara_modul5.

5. Laporan sementara harus diupload pada classroom, jika tidak maka akan terkena pelanggaran dan tidak akan dinilai.



Tugas Praktikum

Langkah Pertama

Membuat koneksi database Oracle dengan Java.

Query

```
package com.outsourcingcompany.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class OracleConn {
    private Connection conn;
    private Statement db;
    private String database =
"muchlasin06941";

    public OracleConn() {
        try {

Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Class Driver
Ditemukan");

            try{
                conn =
DriverManager.getConnection("jdbc:oracle:thin
:@localhost:1521:orcl","muchlasin06941"/*user
name*/, "muchlas/*pass*/");
                System.out.println("Koneksi
Database sukses");
            } catch (SQLException se){
                System.out.println("Koneksi
Database gagal = " + se);
            }
        } catch (ClassNotFoundException err){
            System.out.println("Class Driver
Tidak Ditemukan, Terjadi kesalahan pada :
"+err);
        }
    }
}
```

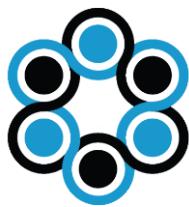


Tugas Praktikum

```
        } catch (SQLException se){
            System.out.println("Koneksi
Database gagal = " + se);
        }
    } catch (ClassNotFoundException err){
        System.out.println("Class Driver
Tidak Ditemukan, Terjadi kesalahan pada :
"+err);
    }
}

public ResultSet GetData(String sql){
    try {
        db =
conn.createStatement	ResultSet.TYPE_SCROLL_IN
SENSITIVE, ResultSet.CONCUR_UPDATABLE);
        return db.executeQuery(sql);
    } catch (SQLException e){
        return null;
    }
}

public int ManipulasiData (String sql){
    try {
        db = conn.createStatement();
        return db.executeUpdate(sql);
    } catch (SQLException e){
        return 0;
    }
}
}
```



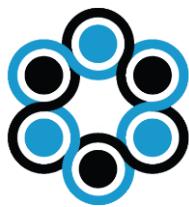
Tugas Praktikum

Screenshot

The screenshot shows the Netbeans IDE's Output window titled "Output - OutsourcingCompany (run)". The window displays the following text:
run:
Class Driver Ditemukan
Koneksi Database sukses
BUILD SUCCESSFUL (total time: 1 second)

Analisa

Source code di atas adalah source code untuk class **OracleConn.java** yang berfungsi untuk membuat koneksi antara IDE Netbeans dengan Java Oracle agar saling terhubung sesuai dengan database yang diinginkan. Selain menggunakan sintaks tertentu pada class ini, perlu insert driver berupa **odbc7.jar** agar sintaks code yang ditulis bisa dieksekusi sesuai kebutuhan atau keinginan.



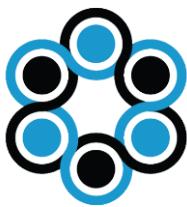
Tugas Praktikum

Langkah Kedua

Membuat class **Employee.java** pada Package: **com.outsourcingcompany.model**.

Query

```
public class Employee {  
    private Integer empId;  
    private String empName;  
    private String empAddr;  
    private String empGender;  
    private String empPhoneNo;  
    private String empEmail;  
  
    public Integer getEmpId() {  
        return empId;  
    }  
  
    public void setEmpId(Integer empId) {  
        this.empId = empId;  
    }  
  
    public String getEmpName() {  
        return empName;  
    }  
  
    public void setEmpName(String empName) {  
        this.empName = empName;  
    }  
  
    public String getEmpAddr() {  
        return empAddr;  
    }  
  
    public void setEmpAddr(String empAddr) {  
        this.empAddr = empAddr;  
    }  
  
    public String getEmpGender() {  
        return empGender;  
    }  
  
    public void setEmpGender(String empGender) {  
        this.empGender = empGender;  
    }  
}
```



Tugas Praktikum

```
public String getEmpPhoneNo() {
    return empPhoneNo;
}

public void setEmpPhoneNo(String empPhoneNo) {
    this.empPhoneNo = empPhoneNo;
}

public String getEmpEmail() {
    return empEmail;
}

public void setEmpEmail(String empEmail)
{
    this.empEmail = empEmail;
}

}
```

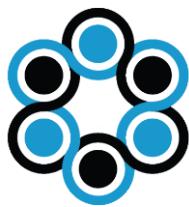
Screenshot

The screenshot shows the NetBeans IDE interface with the Employee.java file open in the editor. The code is identical to the one provided above, defining a class Employee with various getters and setters for attributes like empId, empName, empAddr, empGender, empMail, and empPhoneNo.

```
Source | History | File | New | Open | Recent | Tools | Plugins | Help | Exit | 
12 public class Employee {
13     private Integer empId;
14     private String empName;
15     private String empAddr;
16     private String empGender;
17     private String empPhoneNo;
18     private String empEmail;
19
20     public Integer getEmpId() {
21         return empId;
22     }
23
24     public void setEmpId(Integer empId) {
25         this.empId = empId;
26     }
27
28     public String getEmpName() {
29         return empName;
30     }
31
32     public void setEmpName(String empName) {
33         this.empName = empName;
34     }
35
36     public String getEmpAddr() {
37         return empAddr;
38     }
39
40     public void setEmpAddr(String empAddr) {
41         this.empAddr = empAddr;
42     }
43
44     public String getEmpGender() {
45         return empGender;
46     }
47
48     public void setEmpGender(String empGender) {
49         this.empGender = empGender;
50     }
51
52     public String getEmpPhoneNo() {
53         return empPhoneNo;
54     }
55
56     public void setEmpPhoneNo(String empPhoneNo) {
57         this.empPhoneNo = empPhoneNo;
58     }
59
60     public String getEmpEmail() {
```

Analisa :

Source code di atas adalah source code dari **Employee.java** yang berfungsi sebagai model untuk data pegawai yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: empId, empName, empAddr, empGender, empMail, dan empPhoneNo.



Tugas Praktikum

Langkah Keempat

Membuat class **Client.java** pada Package: **com.outsourcingcompany.model**.

Query

```
public class Client {  
    private Integer clId;  
    private String clName;  
    private String clAddr;  
    private String clEmail;  
    private String clPhoneNo;  
  
    public Integer getClId() {  
        return clId;  
    }  
  
    public void setClId(Integer clId) {  
        this.clId = clId;  
    }  
  
    public String getClName() {  
        return clName;  
    }  
  
    public void setClName(String clName) {  
        this.clName = clName;  
    }  
  
    public String getClAddr() {  
        return clAddr;  
    }  
  
    public void setClAddr(String clAddr) {  
        this.clAddr = clAddr;  
    }  
  
    public String getClEmail() {  
        return clEmail;  
    }  
  
    public void setClEmail(String clEmail) {  
        this.clEmail = clEmail;  
    }  
}
```



Tugas Praktikum

```
public String getClPhoneNo() {
    return clPhoneNo;
}

public void setClPhoneNo(String
clPhoneNo) {
    this.clPhoneNo = clPhoneNo;
}

}
```

Screenshot

The screenshot shows the NetBeans IDE interface with the code editor open. The code is a Java class named Client. It contains private fields for clId, clName, clAddr, clEmail, and clPhoneNo. It has corresponding getter and setter methods for each field. The code is well-organized with proper indentation and line numbers.

```
11  /*
12  * To change this license header, choose License Headers in Project Properties.
13  * To change this template file, choose Tools | Templates
14  * and open the template in the editor.
15  */
16  package client;
17
18  public class Client {
19
20     private Integer clId;
21
22     private String clName;
23
24     private String clAddr;
25
26     private String clEmail;
27
28     private String clPhoneNo;
29
30
31     public Integer getClId() {
32         return clId;
33     }
34
35     public void setClId(Integer clId) {
36         this.clId = clId;
37     }
38
39     public String getClName() {
40         return clName;
41     }
42
43     public void setClName(String clName) {
44         this.clName = clName;
45     }
46
47     public String getClAddr() {
48         return clAddr;
49     }
50
51     public void setClAddr(String clAddr) {
52         this.clAddr = clAddr;
53     }
54
55     public String getClEmail() {
56         return clEmail;
57     }
58
59     public void setClEmail(String clEmail) {
60         this.clEmail = clEmail;
61     }
62
63     public String getClPhoneNo() {
64         return clPhoneNo;
65     }
66
67     public void setClPhoneNo(String clPhoneNo) {
68         this.clPhoneNo = clPhoneNo;
69     }
70
71 }
```

Analisa :

Source code di atas adalah source code dari **Client.java** yang berfungsi sebagai model untuk data client yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: clId, clName, clAddr, clEmail, clPhoneNo.



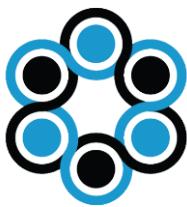
Tugas Praktikum

Langkah Kelima

Membuat class **Job.java** pada Package: **com.outsourcingcompany.model**.

Query

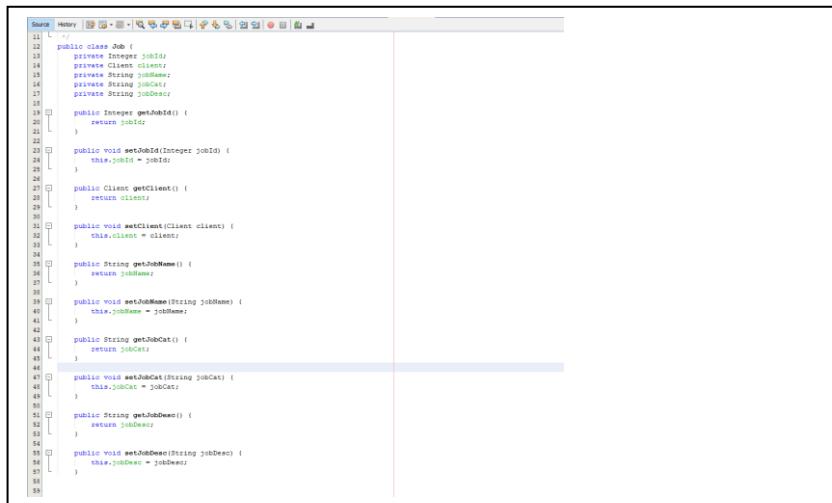
```
public class Job {  
    private Integer jobId;  
    private Client client;  
    private String jobName;  
    private String jobCat;  
    private String jobDesc;  
  
    public Integer getJobId() {  
        return jobId;  
    }  
  
    public void setJobId(Integer jobId) {  
        this.jobId = jobId;  
    }  
  
    public Client getClient() {  
        return client;  
    }  
  
    public void setClient(Client client) {  
        this.client = client;  
    }  
  
    public String getJobName() {  
        return jobName;  
    }  
  
    public void setJobName(String jobName) {  
        this.jobName = jobName;  
    }  
  
    public String getJobCat() {  
        return jobCat;  
    }  
  
    public void setJobCat(String jobCat) {  
        this.jobCat = jobCat;  
    }  
}
```



Tugas Praktikum

```
public String getJobDesc() {  
    return jobDesc;  
}  
  
public void setJobDesc(String jobDesc) {  
    this.jobDesc = jobDesc;  
}  
  
}  
}
```

Screenshot

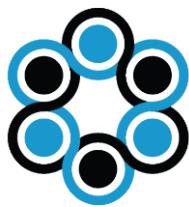


The screenshot shows the NetBeans IDE interface with the source code of the **Job.java** class. The code defines a class with private attributes: jobId, client, jobName, jobCat, and jobDesc. It includes methods for getting and setting each of these attributes, as well as methods to get the job ID and name.

```
11  package com.mycompany.netbeans; // Generated by Oracle IDE  
12  
13  public class Job {  
14      private Integer jobId;  
15      private Client client;  
16      private String jobName;  
17      private String jobCat;  
18      private String jobDesc;  
19  
20      public Integer getJobId() {  
21          return jobId;  
22      }  
23      public void setJobId(Integer jobId) {  
24          this.jobId = jobId;  
25      }  
26  
27      public Client getClient() {  
28          return client;  
29      }  
30      public void setClient(Client client) {  
31          this.client = client;  
32      }  
33  
34      public String getJobName() {  
35          return jobName;  
36      }  
37      public void setJobName(String jobName) {  
38          this.jobName = jobName;  
39      }  
40  
41      public String getJobCat() {  
42          return jobCat;  
43      }  
44      public void setJobCat(String jobCat) {  
45          this.jobCat = jobCat;  
46      }  
47  
48      public String getJobDesc() {  
49          return jobDesc;  
50      }  
51      public void setJobDesc(String jobDesc) {  
52          this.jobDesc = jobDesc;  
53      }  
54  
55  }
```

Analisa :

Source code di atas adalah source code dari **Job.java** yang berfungsi sebagai model untuk data client yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: jobId, client, jobName, jobCat, jobDesc.



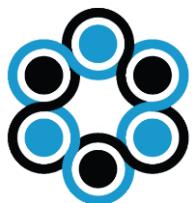
Tugas Praktikum

Langkah Keenam

Membuat class **Project.java** pada Package: **com.outsourcingcompany.model**.

Query

```
public class Project {  
    private Integer prjId;  
    private Job job;  
    private Employee employee;  
    private Date startDate;  
    private Date dueDate;  
  
    public Integer getPrjId() {  
        return prjId;  
    }  
  
    public void setPrjId(Integer prjId) {  
        this.prjId = prjId;  
    }  
  
    public Job getJob() {  
        return job;  
    }  
  
    public void setJob(Job job) {  
        this.job = job;  
    }  
  
    public Employee getEmployee() {  
        return employee;  
    }  
  
    public void setEmployee(Employee  
employee) {  
        this.employee = employee;  
    }  
  
    public Date getStartDate() {  
        return startDate;  
    }  
  
    public void setStartDate(Date startDate)  
{  
        this.startDate = startDate;  
    }  
}
```



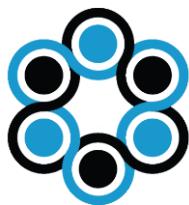
Tugas Praktikum

```
public Date getDueDate() {  
    return dueDate;  
}  
  
public void setDueDate(Date dueDate) {  
    this.dueDate = dueDate;  
}  
}
```

Screenshot

Analisa :

Source code di atas adalah source code dari **Project.java** yang berfungsi sebagai model untuk data client yang akan digunakan sebagai transaksi data oleh Netbeans ke Oracle maupun sebaliknya. Dengan berisi methode **setter** dan **getter** untuk atribut: prjId, empId, jobId, startDate, dueDate.



Tugas Praktikum

Langkah Ketujuh

Membuat class **Controller.java** dengan langkah yang pertama yaitu **import** beberapa **package** serta membuat **constructor**.

Query

```
package com.outsourcingcompany.controller;

import com.outsourcingcompany.model.Employee;
import com.outsourcingcompany.model.Client;
import com.outsourcingcompany.model.Job;
import com.outsourcingcompany.model.Project;
import
com.outsourcingcompany.database.OracleConn;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class Controller {

    OracleConn conn;

    ArrayList<Employee> arrEmployee;
    ArrayList<Client> arrClient;
    ArrayList<Job> arrJob;
    ArrayList<Project> arrProject;

    public Controller() throws SQLException {
        this.conn = new OracleConn();
        this.arrEmployee = new ArrayList<>();
        this.arrClient = new ArrayList<>();
        this.arrJob = new ArrayList<>();
        this.arrProject = new ArrayList<>();
    }
}
```



Tugas Praktikum

Screenshot

```
package com.outsourcingcompany.controller;

import com.outsourcingcompany.model.Employee;
import com.outsourcingcompany.model.Client;
import com.outsourcingcompany.model.Job;
import com.outsourcingcompany.model.Project;
import com.outsourcingcompany.database.OracleConn;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

/**
 * @author Achmad Muchlasin
 */
public class Controller {

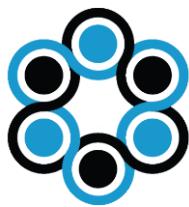
    OracleConn conn;

    ArrayList<Employee> arrEmployee;
    ArrayList<Client> arrClient;
    ArrayList<Job> arrJob;
    ArrayList<Project> arrProject;

    public Controller() throws SQLException {
        this.conn = new OracleConn();
        this.arrEmployee = new ArrayList<>();
        this.arrClient = new ArrayList<>();
        this.arrJob = new ArrayList<>();
        this.arrProject = new ArrayList<>();
    }
}
```

Analisa :

Setelah membuat dan coding di dalam package Model dari **Employee.java**, **Client.java**, **Job.java**, dan **Project.java**. Selanjutnya kita buat class untuk melakukan instruksi transaksi data dari class yang ada di package Model dengan Oracle Database berupa method Create, Read, Update dan Delete. Dapat dilihat di bagian paling atas class ini, perlu import package model, java.sql, dan java.util agar beberapa sintaks code bisa dieksekusi.



Tugas Praktikum

Langkah Kedelapan

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getEmployeeData()**.

Query

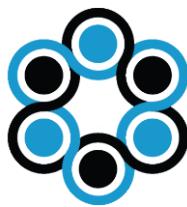
```
public ArrayList<Employee> getEmployeeData() throws  
SQLException {  
    this.arrEmployee.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM EMPLOYEE");  
    while (rs.next()) {  
        Employee emp = new Employee();  
        emp.setEmpId(rs.getInt("ID_EMPLOYEE"));  
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));  
        emp.setEmpAddr(rs.getString("EMPLOYEE_ADDR"));  
        emp.setEmpGender(rs.getString("GENDER"));  
        emp.setEmpPhoneNo(rs.getString("PHONE_NO"));  
        emp.setEmpEmail(rs.getString("EMAIL"));  
        this.arrEmployee.add(emp);  
    }  
    return this.arrEmployee;  
}
```

Screenshot

```
public ArrayList<Employee> getEmployeeData() throws SQLException {  
    this.arrEmployee.clear();  
    ResultSet rs = this.conn.GetData("SELECT * FROM EMPLOYEE");  
    while (rs.next()) {  
        Employee emp = new Employee();  
        emp.setEmpId(rs.getInt("ID_EMPLOYEE"));  
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));  
        emp.setEmpAddr(rs.getString("EMPLOYEE_ADDR"));  
        emp.setEmpGender(rs.getString("GENDER"));  
        emp.setEmpPhoneNo(rs.getString("PHONE_NO"));  
        emp.setEmpEmail(rs.getString("EMAIL"));  
        this.arrEmployee.add(emp);  
    }  
  
    return this.arrEmployee;  
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getEmployeeData()** yang berfungsi untuk membuat mengambil data-data dari **table Employee**.



Tugas Praktikum

Langkah Kesembilan

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getClientData()**.

Query

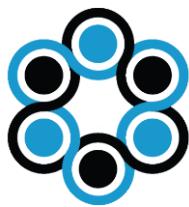
```
public ArrayList<Client> getClientData() throws  
SQLException {  
    this.arrClient.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM CLIENT");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setClId(rs.getInt("ID_CLIENT"));  
        cl.setClName(rs.getString("CLIENT_NAME"));  
        cl.setClAddr(rs.getString("CLIENT_ADDR"));  
        cl.setClEmail(rs.getString("CLIENT_EMAIL"));  
        cl.setClPhoneNo(rs.getString("CLIENT_PHONE_NO"));  
        this.arrClient.add(cl);  
    }  
    return this.arrClient;  
}
```

Screenshot

```
public ArrayList<Client> getClientData() throws SQLException {  
    this.arrClient.clear();  
    ResultSet rs = this.conn.GetData("SELECT * FROM CLIENT");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setClId(rs.getInt("ID_CLIENT"));  
        cl.setClName(rs.getString("CLIENT_NAME"));  
        cl.setClAddr(rs.getString("CLIENT_ADDR"));  
        cl.setClEmail(rs.getString("CLIENT_EMAIL"));  
        cl.setClPhoneNo(rs.getString("CLIENT_PHONE_NO"));  
        this.arrClient.add(cl);  
    }  
  
    return this.arrClient;  
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getClientData()** yang berfungsi untuk membuat mengambil data-data dari **table Client**.



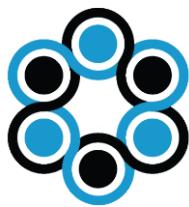
Tugas Praktikum

Langkah Kesepuluh

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getJobData()**.

Query

```
public ArrayList<Job> getJobData() throws  
SQLException {  
    this.arrJob.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM JOB JOIN CLIENT ON JOB.ID_CLIENT =  
CLIENT.ID_CLIENT");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setClId(rs.getInt("ID_CLIENT"));  
        cl.setClName(rs.getString("CLIENT_NAME"));  
        cl.setClAddr(rs.getString("CLIENT_ADDR"));  
        cl.setClEmail(rs.getString("CLIENT_EMAIL"));  
  
        cl.setClPhoneNo(rs.getString("CLIENT_PHONE_NO"));  
  
        Job job = new Job();  
        job.setJobId(rs.getInt("ID_JOB"));  
        job.setClient(cl);  
  
        job.setJobName(rs.getString("JOB_NAME"));  
        job.setJobCat(rs.getString("CATEGORY"));  
        job.setJobDesc(rs.getString("DESCRIPTION"));  
        this.arrJob.add(job);  
    }  
  
    return this.arrJob;  
}
```



Tugas Praktikum

Screenshot

```
public ArrayList<Job> getJobData() throws SQLException {
    this.arrJob.clear();
    ResultSet rs = this.conn.GetData("SELECT * FROM JOB JOIN CLIENT ON JOB.ID_CLIENT = CLIENT.ID_CLIENT");
    while (rs.next()) {
        Client cl = new Client();
        cl.setClientId(rs.getInt("ID_CLIENT"));
        cl.setCName(rs.getString("CLIENT_NAME"));
        cl.setCAddr(rs.getString("CLIENT_ADDR"));
        cl.setCEmail(rs.getString("CLIENT_EMAIL"));
        cl.setCPhoneNo(rs.getString("CLIENT_PHONE_NO"));

        Job job = new Job();
        job.setJobId(rs.getInt("ID_JOB"));
        job.setClient(cl);
        job.setJobName(rs.getString("JOB_NAME"));
        job.setJobCat(rs.getString("CATEGORY"));
        job.setJobDesc(rs.getString("DESCRIPTION"));
        this.arrJob.add(job);
    }

    return this.arrJob;
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getJobData()** yang berfungsi untuk membuat mengambil data-data dari **table Job**.



Tugas Praktikum

Langkah Kesebelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getJobClientData()**.

Query

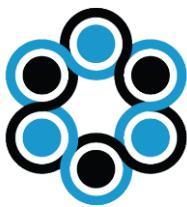
```
public ArrayList<Job> getJobClientData() throws  
SQLException {  
    this.arrJob.clear();  
    ResultSet rs = this.conn.GetData("SELECT *  
FROM JOB_FROM_CLIENT_V");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setCId(rs.getInt("ID_CLIENT"));  
        cl.setName(rs.getString("CLIENT_NAME"));  
        Job job = new Job();  
        job.setId(rs.getInt("ID_JOB"));  
        job.setClient(cl);  
        job.setName(rs.getString("JOB_NAME"));  
        job.setDescription(rs.getString("DESCRIPTION"));  
        this.arrJob.add(job);  
    }  
    return this.arrJob;  
}
```

Screenshot

```
public ArrayList<Job> getJobClientData() throws SQLException {  
    this.arrJob.clear();  
    ResultSet rs = this.conn.GetData("SELECT * FROM JOB_FROM_CLIENT_V");  
    while (rs.next()) {  
        Client cl = new Client();  
        cl.setCId(rs.getInt("ID_CLIENT"));  
        cl.setName(rs.getString("CLIENT_NAME"));  
  
        Job job = new Job();  
        job.setId(rs.getInt("ID_JOB"));  
        job.setClient(cl);  
        job.setName(rs.getString("JOB_NAME"));  
        job.setDescription(rs.getString("DESCRIPTION"));  
        this.arrJob.add(job);  
    }  
  
    return this.arrJob;  
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getJobClientData()** yang berfungsi untuk membuat mengambil data-data dari VIEW yang sudah dibuat antara **table Job** dan **table Client**.



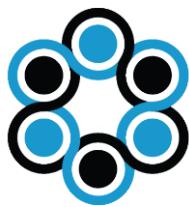
Tugas Praktikum

Langkah Keduabelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **getProjectData()**.

Query

```
public ArrayList<Project> getProjectData() throws  
SQLException {  
    this.arrProject.clear();  
    ResultSet rs = this.conn.GetData("SELECT  
PROJECT.ID_PROJECT, JOB.JOB_NAME,  
EMPLOYEE.EMPLOYEE_NAME, PROJECT.START_DATE,  
PROJECT.END_DATE FROM PROJECT JOIN JOB ON  
PROJECT.ID_JOB = JOB.ID_JOB JOIN EMPLOYEE ON  
PROJECT.ID_EMPLOYEE = EMPLOYEE.ID_EMPLOYEE ORDER BY  
PROJECT.ID_PROJECT ASC");  
    while (rs.next()) {  
        Employee emp = new Employee();  
  
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));  
  
        Job job = new Job();  
  
        job.setJobName(rs.getString("JOB_NAME"));  
  
        Project project = new Project();  
  
        project.setPrjId(rs.getInt("ID_PROJECT"));  
        project.setJob(job);  
        project.setEmployee(emp);  
  
        project.setStartDate(rs.getDate("START_DATE"));  
  
        project.setDueDate(rs.getDate("END_DATE"));  
  
        this.arrProject.add(project);  
    }  
  
    return this.arrProject;  
}
```



Tugas Praktikum

Screenshot

```
public ArrayList<Project> getProjectData() throws SQLException {
    this.arrProject.clear();
    ResultSet rs = this.conn.GetData("SELECT PROJECT.ID_PROJECT, "
        + "JOB.JOB_NAME, EMPLOYEE.EMPLOYEE_NAME, PROJECT.START_DATE, "
        + "PROJECT.END_DATE FROM PROJECT JOIN JOB "
        + "ON PROJECT.ID_JOB = JOB.ID_JOB "
        + "JOIN EMPLOYEE ON PROJECT.ID_EMPLOYEE = EMPLOYEE.ID_EMPLOYEE "
        + "ORDER BY PROJECT.ID_PROJECT ASC");
    while (rs.next()) {
        Employee emp = new Employee();
        emp.setEmpName(rs.getString("EMPLOYEE_NAME"));

        Job job = new Job();
        job.setJobName(rs.getString("JOB_NAME"));

        Project project = new Project();
        project.setPrjId(rs.getInt("ID_PROJECT"));
        project.setJob(job);
        project.setEmployee(emp);
        project.setStartDate(rs.getDate("START_DATE"));
        project.setDueDate(rs.getDate("END_DATE"));

        this.arrProject.add(project);
    }
    return this.arrProject;
}
```

Analisa :

Source code di atas adalah source code dari **Controller.java** pada bagian method **getProjectData()** yang berfungsi untuk membuat mengambil data-data dari **table Project** yang juga menerapkan query JOIN untuk menampilkan **EMPLOYEE_NAME** dan **JOB_NAME** pada **table Employee** dan **table Job..**



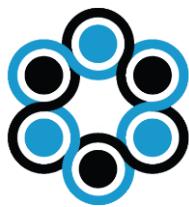
Tugas Praktikum

Langkah Ketigabelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **insertEmployee()**, **insertClient()**, **insertJob()**, **insertProject()**.

Query

```
public void insertEmployee(Employee emp) {
    this.conn.ManipulasiData("INSERT INTO
EMPLOYEE VALUES ('" + emp.getEmpId() + "', '" +
emp.getEmpName() + "', '" + emp.getEmpAddr() + "', '" +
+ emp.getEmpGender() + "', '" + emp.getEmpPhoneNo() + "
', '" + emp.getEmpEmail() + "')");
}
public void insertClient(Client cl) {
    this.conn.ManipulasiData("INSERT INTO CLIENT
VALUES ('" + cl.getClId() + "', '" + cl.getClName() +
"', '" + cl.getClAddr() + "', '" + cl.getClEmail() +
"', '" + cl.getClPhoneNo() + "')");
}
public void insertJob(Job job) {
    this.conn.ManipulasiData("INSERT INTO JOB
VALUES ('" + job.getJobId() + "', '" +
job.getClient().getClId() + "', '" + job.getJobName() +
"', '" + job.getJobCat() + "', '" +
job.getJobDesc() + "')");
}
public void insertProject(Project project) {
    try {
        String startDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getStar
tDate());
        String dueDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getDueD
ate());
        this.conn.ManipulasiData("INSERT INTO
PROJECT VALUES (ID_PROJECT.NEXTVAL, " +
project.getJob().getJobId() + ", " +
project.getEmployee().getEmpId() + ", TO_DATE('" +
startDate + "', 'dd/MM/yyyy'), TO_DATE('" + dueDate + "
', 'dd/MM/yyyy'))");
    } catch (Exception e) {
        System.out.println(e);
    }
}
```



Tugas Praktikum

Screenshot

```
public void insertEmployee(Employee emp) {
    this.conn.ManipulasiData("INSERT INTO EMPLOYEE VALUES ('" + emp.getEmpId() + "', '" +
        + emp.getEmpName() + "', '" + emp.getEmpAddr() + "', '" +
        + emp.getEmpGender() + "', '" + emp.getEmpPhoneNo() + "', '" +
        + emp.getEmpEmail() + "')");
}

public void insertClient(Client cl) {
    this.conn.ManipulasiData("INSERT INTO CLIENT VALUES ('" + cl.getCliId() + "', '" +
        + cl.getClName() + "', '" + cl.getClAddr() + "', '" +
        + cl.getClEmail() + "', '" +
        + cl.getClPhoneNo() + "')");
}

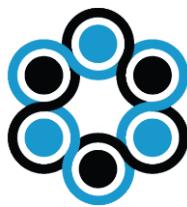
public void insertJob(Job job) {
    this.conn.ManipulasiData("INSERT INTO JOB VALUES ('" + job.getJobId() + "', '" +
        + job.getClient().getCliId() + "', '" + job.getJobName() + "', '" +
        + job.getJobCat() + "', '" +
        + job.getJobDesc() + "')");
}

public void insertProject(Project project) {
    try {
        String startDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getStartDate());
        String dueDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getDueDate());
        this.conn.ManipulasiData("INSERT INTO PROJECT VALUES (ID_PROJECT.NEXTVAL, " +
            + project.getJob().getJobId() + ", " +
            + project.getEmployee().getEmpId() +
            + ", TO_DATE('" + startDate + "', 'dd/MM/yyyy'), TO_DATE('" + dueDate + "', 'dd/MM/yyyy'))");
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

Analisa :

Source code di atas adalah source code **Controller.java** pada bagian method yang digunakan untuk melakukan eksekusi CREATE / INSERT DATA pada table Employee, Client, Job, dan Project. Bisa dilihat pada gambar terdapat beberapa method, yaitu :

- insertEmployee(Employee emp)
- insertClient(Clienet cl)
- insertJob(Job job)
- insertProject(Project project)



Tugas Praktikum

Langkah Keempatbelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **updateEmployee()**, **updateClient()**, **updateJob()**, **updateProject()**.

Query

```
public void updateEmployee(Employee emp) {
    this.conn.ManipulasiData("UPDATE EMPLOYEE SET
EMPLOYEE_NAME = '" + emp.getEmpName()
        + "', EMPLOYEE_ADDR = '" +
emp.getEmpAddr()
        + "', GENDER = '" +
emp.getEmpGender()
        + "', PHONE_NO = '" +
emp.getEmpPhoneNo()
        + "', EMAIL = '" + emp.getEmpEmail()
        + "' WHERE ID_EMPLOYEE = " +
emp.getEmpId());
}

public void updateClient(Client cl) {
    this.conn.ManipulasiData("UPDATE CLIENT SET
ID_CLIENT = " + cl.getClId()
        + ", CLIENT_NAME = '" +
cl.getClName()
        + "', CLIENT_ADDR = '" +
cl.getClAddr()
        + "', CLIENT_EMAIL = '" +
cl.getClEmail()
        + "', CLIENT_PHONE_NO = '" +
cl.getClPhoneNo()
        + "' WHERE ID_CLIENT = " +
cl.getClId());
}

public void updateJob(Job j) {
    this.conn.ManipulasiData("UPDATE JOB SET
ID_CLIENT = " + j.getClient().getClId()
        + ", JOB_NAME = '" + j.getJobName()
        + "', CATEGORY = '" + j.getJobCat()
        + "', DESCRIPTION = '" +
j.getJobDesc()
        + "' WHERE ID_JOB = " +
j.getJobId());
```



Tugas Praktikum

```
}
```

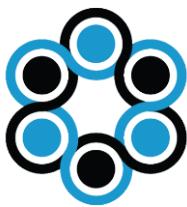
```
public void updateProject(Project project) {
    String startDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getStar
tDate());
    String dueDate = new
SimpleDateFormat("dd/MM/yyyy").format(project.getDueD
ate());
    this.conn.ManipulasiData("UPDATE PROJECT SET
ID_JOB = " + project.getJob().getJobId()
        + ", ID_EMPLOYEE = " +
project.getEmployee().getEmpId()
        + ", START_DATE = TO_DATE('" +
startDate
        + "', 'dd/MM/yyyy'), END_DATE =
TO_DATE('" + dueDate
        + "', 'dd/MM/yyyy') WHERE ID_PROJECT
= " + project.getPrjId());
}
```

Screenshot

```
public void updateEmployee(Employee emp) {
    this.conn.ManipulasiData("UPDATE EMPLOYEE SET EMPLOYEE_NAME = '" + emp.getEmpName()
        + "', EMPLOYEE_ADDR = '" + emp.getEmpAddr()
        + "', GENDER = '" + emp.getEmpGender()
        + "', PHONE_NO = '" + emp.getEmpPhoneNo()
        + "', EMAIL = '" + emp.getEmpEmail()
        + "' WHERE ID_EMPLOYEE = " + emp.getEmpId());
}

public void updateClient(Client cl) {
    this.conn.ManipulasiData("UPDATE CLIENT SET ID_CLIENT = " + cl.getCliId()
        + ", CLIENT_NAME = '" + cl.getClName()
        + "', CLIENT_ADDR = '" + cl.getClAddr()
        + "', CLIENT_EMAIL = '" + cl.getClEmail()
        + "', CLIENT_PHONE_NO = '" + cl.getClPhoneNo()
        + "' WHERE ID_CLIENT = " + cl.getClId());
}

public void updateJob(Job j) {
    this.conn.ManipulasiData("UPDATE JOB SET ID_CLIENT = " + j.getClient().getClId()
        + ", JOB_NAME = '" + j.getJobName()
        + "', CATEGORY = '" + j.getJobCat()
        + "', DESCRIPTION = '" + j.getJobDesc()
        + "' WHERE ID_JOB = " + j.getJobId());
}
```



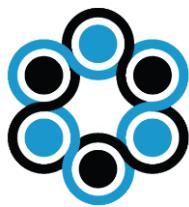
Tugas Praktikum

```
public void updateProject(Project project) {  
    String startDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getStartDate());  
    String dueDate = new SimpleDateFormat("dd/MM/yyyy").format(project.getDueDate());  
    this.conn.ManipulasiData("UPDATE PROJECT SET ID_JOB = " + project.getJob().getJobId()  
        + ", ID_EMPLOYEE = " + project.getEmployee().getEmpId()  
        + ", START_DATE = TO_DATE('" + startDate  
        + "', 'dd/MM/yyyy'), END_DATE = TO_DATE('" + dueDate  
        + "', 'dd/MM/yyyy') WHERE ID_PROJECT = " + project.getPrjId());  
}
```

Analisa :

Source code di atas adalah source code **Controller.java** pada bagian method yang digunakan untuk melakukan eksekusi UPDATE DATA pada table Employee, Client, Job, dan Project. Bisa dilihat pada gambar terdapat beberapa method, yaitu:

- updateEmployee(Employee emp)
- updateClient(Clienet cl)
- updateJob(Job j)
- updateProject(Project project)



Tugas Praktikum

Langkah Kelimabelas

Masih di class **Controller.java** dengan langkah yang selanjutnya yaitu membuat method **deleteEmployee()**, **deleteClient()**, **deleteJob()**, **deleteProject()**.

Query

```
public void deleteEmployee(Integer idEmp) {
    this.conn.ManipulasiData("DELETE FROM
EMPLOYEE WHERE ID_EMPLOYEE = " + idEmp);
}
public void deleteClient(Integer idClient) {
    this.conn.ManipulasiData("DELETE FROM CLIENT
WHERE ID_CLIENT = " + idClient);
}
public void deleteJob(Integer idJob) {
    this.conn.ManipulasiData("DELETE FROM JOB
WHERE ID_JOB = " + idJob);
}
public void deleteProject(Integer Id) {
    this.conn.ManipulasiData("DELETE FROM PROJECT
WHERE ID_PROJECT = " + Id);
}
```

Screenshot

```
public void deleteEmployee(Integer idEmp) {
    this.conn.ManipulasiData("DELETE FROM EMPLOYEE WHERE ID_EMPLOYEE = " + idEmp);
}

public void deleteClient(Integer idClient) {
    this.conn.ManipulasiData("DELETE FROM CLIENT WHERE ID_CLIENT = " + idClient);
}

public void deleteJob(Integer idJob) {
    this.conn.ManipulasiData("DELETE FROM JOB WHERE ID_JOB = " + idJob);
}

public void deleteProject(Integer Id) {
    this.conn.ManipulasiData("DELETE FROM PROJECT WHERE ID_PROJECT = " + Id);
}
```

Analisa :

Source code di atas adalah source code **Controller.java** pada bagian method yang digunakan untuk melakukan eksekusi **DELETE DATA** pada table Employee, Client, Job, dan Project.



Tugas Praktikum

Langkah Keenambelas

Pindah ke class **OutsourcingCompany.java** , class tersebut adalah main class pada aplikasi ini.

Query

```
package com.outsourcingcompany.main;

import com.outsourcingcompany.view.*;
import java.sql.SQLException;
import java.text.ParseException;

public class OutsourcingCompany {
    public static void main(String[] args) throws
SQLException, ParseException {
        // TODO code application logic here
        new LoginView().show();
    }
}
```

Screenshot

```
package com.outsourcingcompany.main;

import com.outsourcingcompany.view.*;
import java.sql.SQLException;
import java.text.ParseException;

public class OutsourcingCompany {
    public static void main(String[] args) throws SQLException, ParseException {
        // TODO code application logic here
        new LoginView().show();
    }
}
```

Analisa :

Source code di atas adalah source code dari **OutsourcingCompany.java** yang merupakan main class dari aplikasi ini. Class ini berfungsi untuk memanggil class **LoginView.java** pada package View yang berisi semua class JFrame / GUI.



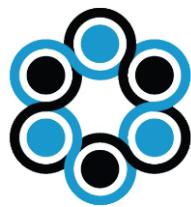
Tugas Praktikum

Langkah Ketujuhbelas

Membuat class JFrame dengan nama **LoginView.java**.

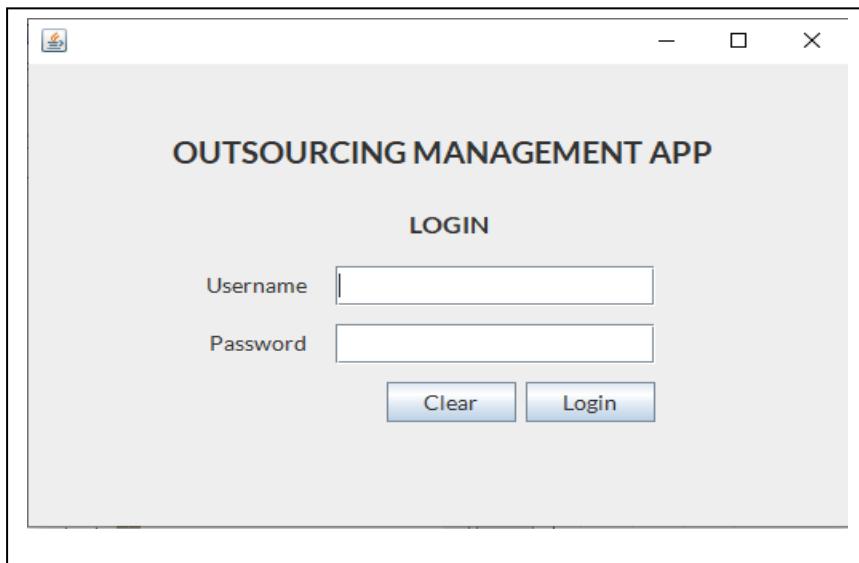
Query

```
private void  
btnLoginOkActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String username =  
txtUsername.getText().trim();  
    String password =  
txtPassword.getText().trim();  
  
    if(username.equals("admin") &&  
password.equals("admin123")) {  
        try {  
            new HomeAdminView().show();  
            setVisible(false);  
        } catch (SQLException ex) {  
  
Logger.getLogger(LoginView.class.getName()).log(Level.SEVERE, null, ex);  
        }  
    } else if(username.equals("hrd") &&  
password.equals("hrd123")) {  
        try {  
            new ProjectView().show();  
            setVisible(false);  
        } catch (SQLException ex) {  
  
Logger.getLogger(LoginView.class.getName()).log(Level.SEVERE, null, ex);  
        }  
    } else {  
  
JOptionPane.showMessageDialog(null, "WRONG  
PASSWORD!");  
        setVisible(false);  
    }  
}
```



Tugas Praktikum

Screenshot



Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **LoginView.java** yang berfungsi untuk membuat halaman login. Username yang disediakan adalah sebagai berikut:

Login Admin

Username : **admin**

Password : **admin123**

Login HRD

Username : **hrd**

Password : **hrd123**



Tugas Praktikum

Langkah Kedelapanbelas

Membuat class JFrame dengan nama **HomeAdminView.java**.

Query

```
private void
btnEntryPegawaiActionPerformed(java.awt.event.
ActionEvent evt) {
    // TODO add your handling code here:
    try {
        new EntryEmployeeView().show();
    } catch (SQLException e) {

Logger.getLogger(LoginView.class.getName()).lo
g(Level.SEVERE, null, e);
    }
}

private void
btnEntryClientActionPerformed(java.awt.event.A
ctionEvent evt) {
    // TODO add your handling code here:
    try {
        new EntryClientView().show();
    } catch (SQLException e) {

Logger.getLogger(LoginView.class.getName()).lo
g(Level.SEVERE, null, e);
    }
}

private void
btnEntryJobActionPerformed(java.awt.event.Acti
onEvent evt) {
    // TODO add your handling code here:
    try {
        new EntryJobView().show();
    } catch (SQLException e) {

Logger.getLogger(LoginView.class.getName()).lo
g(Level.SEVERE, null, e);
    }
}
```



Tugas Praktikum

```
private void
btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int ans =
JOptionPane.showOptionDialog(this,
        "Do you want to logout?",
        "Logout",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE,
null, null, null);

    if (ans == JOptionPane.YES_OPTION) {

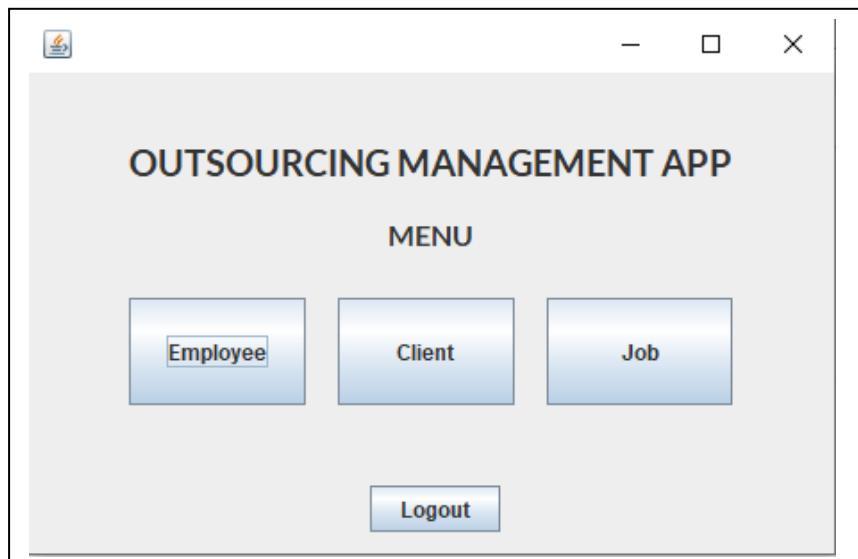
JOptionPane.showMessageDialog(null, "Logout
succeeded!");
try {
    new LoginView().show();
    setVisible(false);

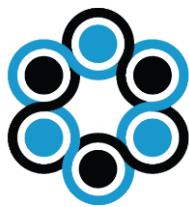
} catch (SQLException ex) {

Logger.getLogger(HomeAdminView.class.getName()
).log(Level.SEVERE, null, ex);
}

    }
}
```

Screenshot





Tugas Praktikum

Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **HomeAdminView.java** yang berfungsi sebagai menu bagi Admin apakah ingin melakukan transaksi data terhadap Employee, Client, atau Job. Terdapat button untuk logout juga jika admin sudah atau tidak akan melakukan transaksi data terhadap tiga kategori tersebut.



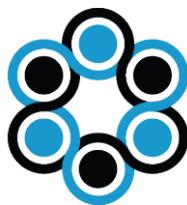
Tugas Praktikum

Langkah Kesembilanbelas

Membuat class JFrame dengan nama **EntryEmployeeView.java**.

Query

```
private void  
btnAddEmpActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try {  
        Employee emp = new Employee();  
  
        emp.setEmpId(Integer.parseInt(this.txtEmpId.getText()));  
  
        emp.setEmpName(this.txtEmpName.getText());  
  
        emp.setEmpAddr(this.txtEmpAddr.getText());  
  
        emp.setEmpGender(this.cbEmpGender.getSelectedItem().toString());  
  
        emp.setEmpPhoneNo(this.txtEmpPhone.getText());  
  
        emp.setEmpEmail(this.txtEmpEmail.getText());  
  
        this.controller.insertEmployee(emp);  
  
        JOptionPane.showMessageDialog(null, "Employee  
added!");  
        this.showTableEmployeeData();  
    } catch (SQLException ex) {  
        Logger.getLogger(EntryEmployeeView.class.getName()).log(Level.SEVERE, null, ex);  
    }  
  
}  
  
private void  
btnUpdateEmpActionPerformed(java.awt.event.Act
```



Tugas Praktikum

```
private void
btnUpdateEmpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbEmployeeData.getValueAt(this.tbEmployee
Data.getSelectedRow(), 0).toString();
        Employee emp = new Employee();

        emp.setEmpId(Integer.parseInt(selected));

        emp.setEmpName(this.txtEmpName.getText());

        emp.setEmpAddr(this.txtEmpAddr.getText());

        emp.setEmpGender(this.cbEmpGender.getSelectedItem()
.toString());

        emp.setEmpPhoneNo(this.txtEmpPhone.getText());

        emp.setEmpEmail(this.txtEmpEmail.getText());

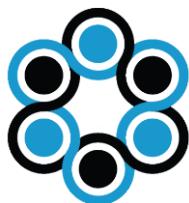
        this.controller.updateEmployee(emp);

        JOptionPane.showMessageDialog(null, "Employee
updated!");
        this.showTableEmployeeData();

    } catch (SQLException e) {

        Logger.getLogger(EntryEmployeeView.class.getName())
.log(Level.SEVERE, null, e);
    }
}

private void
btnDelEmpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbEmployeeData.getValueAt(this.tbEmployee
Data.getSelectedRow(), 0).toString();
    }
}
```



Tugas Praktikum

```
Employee emp = new Employee();
emp.setEmpId(Integer.parseInt(selected));

emp.setEmpName(this.txtEmpName.getText());
emp.setEmpAddr(this.txtEmpAddr.getText());
emp.setEmpGender(this.cbEmpGender.getSelectedItem().toString());
emp.setEmpPhoneNo(this.txtEmpPhone.getText());
emp.setEmpEmail(this.txtEmpEmail.getText());
this.controller.updateEmployee(emp);

JOptionPane.showMessageDialog(null, "Employee updated!");
this.showTableEmployeeData();

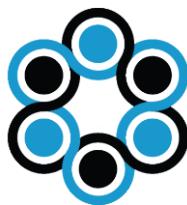
} catch (SQLException e) {
Logger.getLogger(EntryEmployeeView.class.getName()).log(Level.SEVERE, null, e);
}

private void
btnDelEmpActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
try {
String selected =
this.tbEmployeeData.getValueAt(this.tbEmployeeData.getSelectedRow(), 0).toString();

this.controller.deleteEmployee(Integer.parseInt(selected));

JOptionPane.showMessageDialog(null, "Employee deleted!");
this.showTableEmployeeData();

} catch (SQLException ex) {
```



Tugas Praktikum

```
Logger.getLogger(EntryEmployeeView.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void
tbEmployeeDataMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    this.txtEmpId.setEditable(false);

    this.txtEmpId.setText(this.tbEmployeeData.getValueAt(this.tbEmployeeData.getSelectedRow(),
0).toString());
}
```

Screenshot

The screenshot shows a Java Swing application window titled "OUTSOURCING MANAGEMENT APP". The window has a title bar with standard minimize, maximize, and close buttons. Inside, there is a panel for "Employee Data Entry" containing input fields for Employee ID, Employee Name, Employee Address, Gender (a dropdown menu), Phone No., and Email, along with "Add", "Delete", and "Update" buttons. To the right of this panel is a table titled "EMPLOYEE..". The table has columns for EMPLOYEE.., NAME, ADDRESS, GENDER, PHONE NO, and EMAIL. It lists 15 rows of employee data. The data is as follows:

EMPLOYEE..	NAME	ADDRESS	GENDER	PHONE NO	EMAIL
106	Maya	Medaeng	P	1066666	maya@gma...
107	Aga	Waru	L	1077777	aga@gmai...
108	Nara	Sidoarjo	P	1088888	nara@gma...
109	Moka	Candi	L	1099999	moka@gm...
110	Tara	Tanggulan...	L	1100000	tara@gmai...
111	Roni	Sedati	L	1111111	roni@gmai...
112	Yuna	Waru	P	1222222	yuna@gma...
113	Rainy	Sidoarjo	P	1333333	rainy@gma...
114	Moza	Candi	L	1444444	moza@gm...
115	Taro	Tanggulan...	L	1555555	taro@gmai...
101	Muchias	Dusun Dur...	L	8124186	muchias@...
102	Sterling	Surabaya S...	L	8225045	sterling@g...
103	Narendra	Taman Bu...	L	81110145	narendra@...
104	Nurani	Rungkut	P	9993331	nurani@ns...

Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **EntryEmployeeView.java** yang berfungsi sebagai menu bagi Admin untuk melakukan transaksi data (create, update, dan delete) pada **table Employee** yang ada pada database oracle.



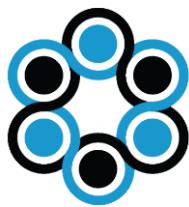
Tugas Praktikum

Langkah Keduapuluhan

Membuat class JFrame dengan nama **EntryClientView.java**.

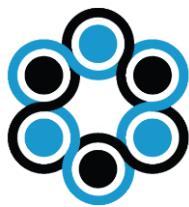
Query

```
private void  
btnAddClientActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try {  
        Client cl = new Client();  
  
        cl.setClId(Integer.parseInt(this.txtClientId.getText()));  
  
        cl.setClName(this.txtClientName.getText());  
  
        cl.setClAddr(this.txtClientAddr.getText());  
  
        cl.setClEmail(this.txtClientEmail.getText());  
  
        cl.setClPhoneNo(this.txtClientPhone.getText())  
;  
        this.controller.insertClient(cl);  
        this.showTableClientData();  
  
    } catch (SQLException ex) {  
  
        Logger.getLogger(EntryClientView.class.getName())  
        .log(Level.SEVERE, null, ex);  
    }  
}  
  
private void  
btnUpdateClActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    try {  
        String selected =  
this.tbClientData.getValueAt(this.tbClientData  
.getSelectedRow(), 0).toString();  
        Client cl = new Client();  
  
        cl.setClId(Integer.parseInt(selected));  
    }
```



Tugas Praktikum

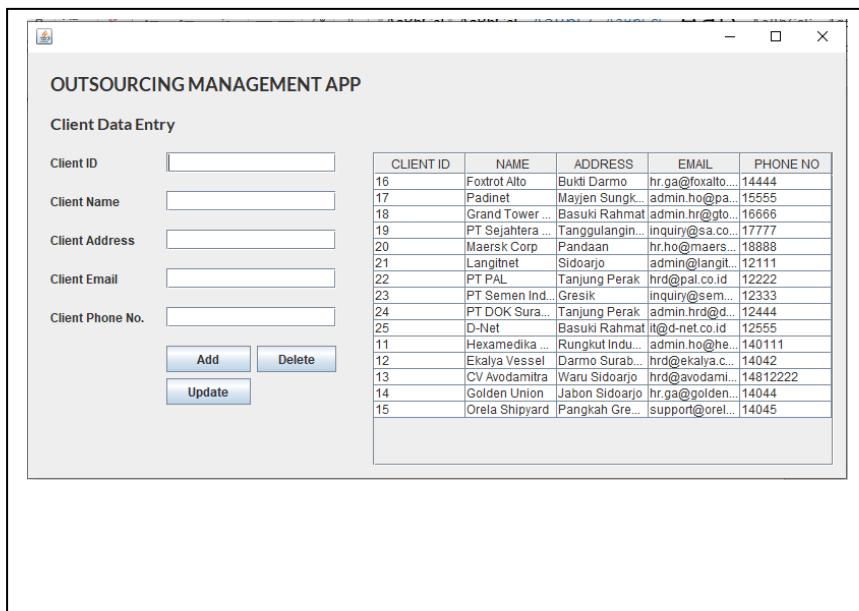
```
cl.setClName(this.txtClientName.getText());  
  
cl.setClAddr(this.txtClientAddr.getText());  
  
cl.setClEmail(this.txtClientEmail.getText());  
  
cl.setClPhoneNo(this.txtClientPhone.getText())  
;  
        this.controller.updateClient(cl);  
        this.showTableClientData();  
  
    } catch (SQLException e) {  
  
        Logger.getLogger(EntryClientView.class.getName()  
()).log(Level.SEVERE, null, e);  
    }  
}  
  
private void  
btnDelClActionPerformed(java.awt.event.ActionEvent  
evt) {  
    // TODO add your handling code here:  
    try {  
        String selected =  
this.tbClientData.getValueAt(this.tbClientData  
.getSelectedRow(), 0).toString();  
  
this.controller.deleteClient(Integer.parseInt(  
selected));  
        this.showTableClientData();  
  
    } catch (SQLException e) {  
  
        Logger.getLogger(EntryClientView.class.getName()  
()).log(Level.SEVERE, null, e);  
    }  
}  
  
private void  
tbClientDataMouseClicked(java.awt.event.MouseE  
vent evt) {  
    // TODO add your handling code here:  
    this.txtClientId.setEditable(false);  
  
this.txtClientId.setText(this.tbClientData.get  
ValueAt(this.tbClientData.getSelectedRow(),
```



Tugas Praktikum

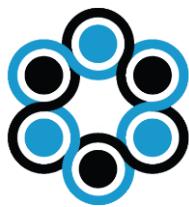
```
private void  
tbClientDataMouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    this.txtClientId.setEditable(false);  
  
    this.txtClientId.setText(this.tbClientData.getValueAt(this.tbClientData.getSelectedRow(),  
0).toString());  
}
```

Screenshot



Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **EntryClientView.java** yang berfungsi sebagai menu bagi Admin untuk melakukan transaksi data (create, update, dan delete) pada **table Client** yang ada pada database oracle.



Tugas Praktikum

Langkah Keduapuluhsatu

Membuat class JFrame dengan nama **EntryJobView.java**.

Query

```
private void
btnAddJobActionPerformed(java.awt.event.ActionEvent
Event evt) {
    // TODO add your handling code here:
    try {
        Job job = new Job();

        job.setJobId(Integer.parseInt(this.txtJobId.ge
tText()));

        job.setClient(this.controller.getClientData().get(this.cbClientId.getSelectedIndex()));

        job.setJobName(this.txtJobName.getText());

        job.setJobCat(this.cbJobCat.getSelectedItem().to
String());

        job.setJobDesc(this.txtJobDesc.getText());
        this.controller.insertJob(job);

        JOptionPane.showMessageDialog(null, "Job
added!");

        this.showTableJobData();

    } catch (SQLException e) {

Logger.getLogger(EntryJobView.class.getName())
.log(Level.SEVERE, null, e);
    }
}

private void
btnUpdJobActionPerformed(java.awt.event.ActionEvent
Event evt) {
    // TODO add your handling code here:
```



Tugas Praktikum

```
private void
btnUpdJobActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    String selected =
this.tbJobData.getValueAt(this.tbJobData.getSelectedRow(), 0).toString();
    Job j = new Job();

    try {

j.setJobId(Integer.parseInt(selected));

j.setClient(this.controller.getClientData().get(this.cbClientId.getSelectedIndex()));

j.setJobName(this.txtJobName.getText());

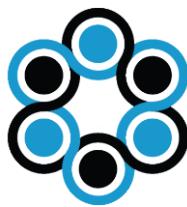
j.setJobCat(this.cbJobCat.getSelectedItem().toString());

j.setJobDesc(this.txtJobDesc.getText());
        this.controller.updateJob(j);

JOptionPane.showMessageDialog(null, "Job
Updated!");
        this.showTableJobData();
    } catch (SQLException ex) {

Logger.getLogger(EntryJobView.class.getName())
.log(Level.SEVERE, null, ex);
    }

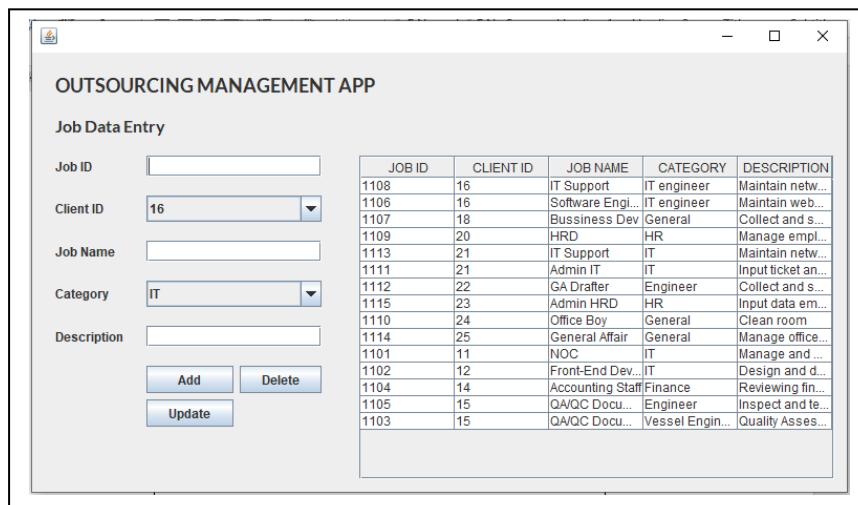
    private void
btnDelJobActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbJobData.getValueAt(this.tbJobData.getSelectedRow(), 0).toString();
```



Tugas Praktikum

```
this.controller.deleteJob(Integer.parseInt(selected));
        JOptionPane.showMessageDialog(null,
"Job Deleted!");
        this.showTableJobData();
    } catch (SQLException ex) {
Logger.getLogger(EntryJobView.class.getName())
.log(Level.SEVERE, null, ex);
    }
}
private void
tbJobDataMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    this.txtJobId.setEditable(false);
this.txtJobId.setText(this.tbJobData.getValueAt(this.tbJobData.getSelectedRow(),
0).toString());
}
}
```

Screenshot



Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **EntryJobView.java** yang berfungsi sebagai menu bagi Admin untuk melakukan transaksi data (create, update, dan delete) pada **table Job** yang ada pada database oracle.



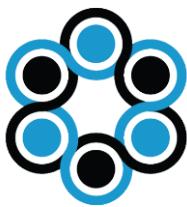
Tugas Praktikum

Langkah Keduapuluuhdua

Membuat class JFrame dengan nama **ProjectView.java**.

Query

```
private void  
btnAddActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    Project project = new Project();  
    try {  
  
        project.setJob(this.assignment.getJobData().ge  
t(this.cbJobId.getSelectedIndex()));  
  
        project.setEmployee(this.assignment.getEmploye  
eData().get(this.cbEmpId.getSelectedIndex()));  
        project.setStartDate(new  
SimpleDateFormat("dd/MM/yyyy").parse(this.txtS  
tartDate.getText()));  
        project.setDueDate(new  
SimpleDateFormat("dd/MM/yyyy").parse(this.txtD  
ueDate.getText()));  
  
        this.assignment.insertProject(project);  
  
        JOptionPane.showMessageDialog(null, "Project  
added!");  
        this.showTableProject();  
    } catch (SQLException | ParseException  
err) {  
    System.out.println(err);  
}  
  
}  
  
private void  
tblEmployeeListMouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
}  
}
```



Tugas Praktikum

```
private void
tbOutstandingJobMouseClicked(java.awt.event.Mo
useEvent evt) {
    // TODO add your handling code here:
}

private void
btnUpdateActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
    try {
        String selectedRow =
this.tbOnProgressProject.getValueAt(this.tbOnP
rogressProject.getSelectedRow(),
0).toString();
        Project project = new Project();

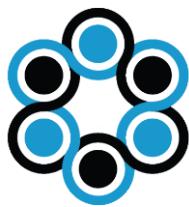
        project.setPrjId(Integer.parseInt(selectedRow)
);

        project.setJob(this.assignment.getJobData().ge
t(this.cbJobId.getSelectedIndex()));

        project.setEmployee(this.assignment.getEmploye
eData().get(this.cbEmpId.getSelectedIndex()));
        project.setStartDate(new
SimpleDateFormat("dd/MM/yyyy").parse(this.txtS
tarDate.getText()));
        project.setDueDate(new
SimpleDateFormat("dd/MM/yyyy").parse(this.txtD
ueDate.getText()));

        this.assignment.updateProject(project);

        JOptionPane.showMessageDialog(null, "Project
updated!");
        this.showTableProject();
    } catch (SQLException | ParseException
err) {
        System.out.println(err);
    }
}
```



Tugas Praktikum

```
private void
btnDelActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String selected =
this.tbOnProgressProject.getValueAt(this.tbOnP
rogressProject.getSelectedRow(),
0).toString();

        this.assignment.deleteProject(Integer.parseInt
(selected));
JOptionPane.showMessageDialog(null, "Project
deleted!");
        this.showTableProject();
    } catch (NumberFormatException | 
SQLException e) {

Logger.getLogger(ProjectView.class.getName()).
log(Level.SEVERE, null, e);
    }
}

private void
btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int ans =
JOptionPane.showOptionDialog(this,
        "Do you want to logout?",
        "Logout",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE,
null, null, null);

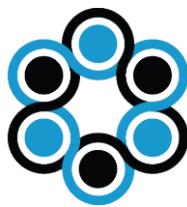
    if (ans == JOptionPane.YES_OPTION) {

JOptionPane.showMessageDialog(null, "You have
been logged out!");
        try {
            new LoginView().show();
            setVisible(false);

        } catch (SQLException ex) {

Logger.getLogger(HomeAdminView.class.getName())

```

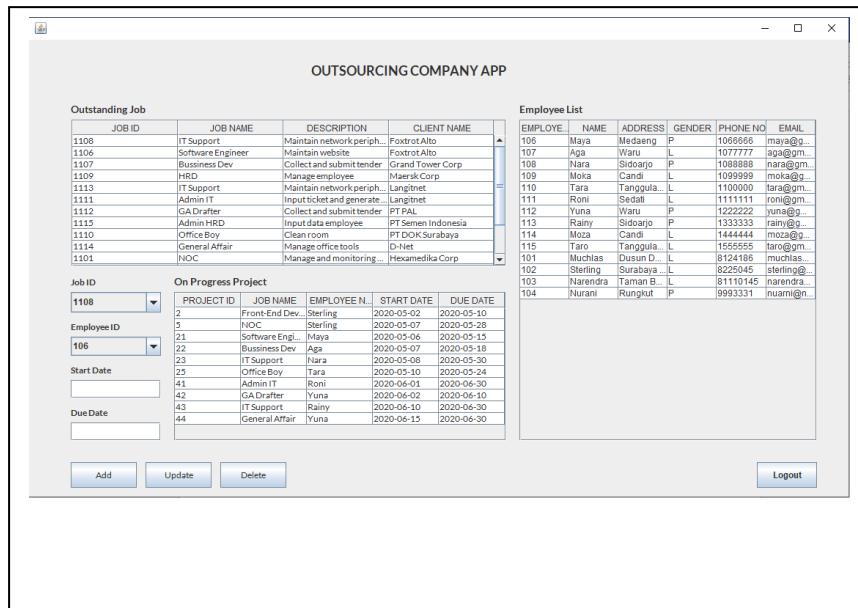


Tugas Praktikum

```
    ).log(Level.SEVERE, null, ex);
}

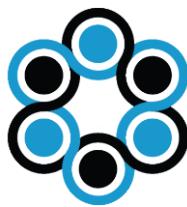
}
```

Screenshot



Analisa :

Pada gambar di atas adalah design dari class JFrame dengan nama **ProjectView.java** yang berfungsi sebagai menu bagi HRD untuk melakukan transaksi data (create, update, dan delete) pada **table Project** yang ada pada database oracle.



Laporan Sementara

Soal Praktikum

1. Buat koneksi database Oracle dengan Java/Web berdasarkan project masing-masing.
2. Buatlah CRUD pada salah satu tabel (boleh menggunakan GUI, Console atau Web).
3. Terapkan salah satu konsep JOIN dan tampilkan pada project Anda.
4. Terapkan salah satu konsep VIEW dan tampilkan pada project Anda.

Ketentuan :

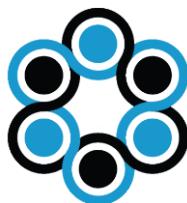
1. Gunakan SQLPlus pada terminal.
2. Kerjakan dalam waktu 150 Menit.

Petunjuk :

1. Bentuk penulisan laporan sementara sesuai template yang ada.
2. Source code dan soal diketik sesuai format yang ada.
3. Upload laporan sementara di classroom dengan batas waktu yang telah ditentukan.
4. Format penamaan upload file :

NPM_NamaPraktikan_LaporanSementara_modul5.

5. Laporan sementara harus diupload pada classroom, jika tidak maka akan terkena pelanggaran dan tidak akan dinilai.



Laporan Sementara

Langkah Pertama

Membuat koneksi database Oracle dengan Java.

Query

```
/*
 * To change this license header, choose
 License Headers in Project Properties.
 * To change this template file, choose Tools
 | Templates
 * and open the template in the editor.
 */
package com.outsourcingcompany.database;

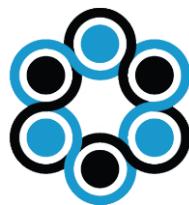
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 *
 * @author Achmad Muchlasin
 */
public class OracleConn {
    private Connection conn;
    private Statement db;
    private String database =
"muchlasin06941";

    public OracleConn() {
        try {

Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("Class Driver
Ditemukan");

            try{
                conn =
DriverManager.getConnection("jdbc:oracle:thin
:@localhost:1521:orcl","muchlasin06941/*user
name*/, "muchlas/*pass*/");
                System.out.println("Koneksi
Database sukses");
            }
        }
    }
}
```



Laporan Sementara

```
        } catch (SQLException se){
            System.out.println("Koneksi
Database gagal = " + se);
        }
    } catch (ClassNotFoundException err){
        System.out.println("Class Driver
Tidak Ditemukan, Terjadi kesalahan pada :
"+err);
    }
}

public ResultSet GetData(String sql){
    try {
        db =
conn.createStatement	ResultSet.TYPE_SCROLL_IN
SENSITIVE, ResultSet.CONCUR_UPDATABLE);
        return db.executeQuery(sql);
    } catch (SQLException e){
        return null;
    }
}

public int ManipulasiData (String sql){
    try {
        db = conn.createStatement();
        return db.executeUpdate(sql);
    } catch (SQLException e){
        return 0;
    }
}
}
```



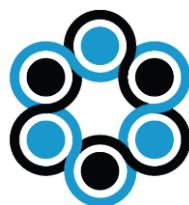
Laporan Sementara

Screenshot

```
Output - OutsourcingCompany (run) X
run:
Class Driver Ditemukan
Koneksi Datatabase sukses
BUILD SUCCESSFUL (total time: 1 second)
```

Analisa

(Ketik hasil analisa disini)



Laporan Sementara

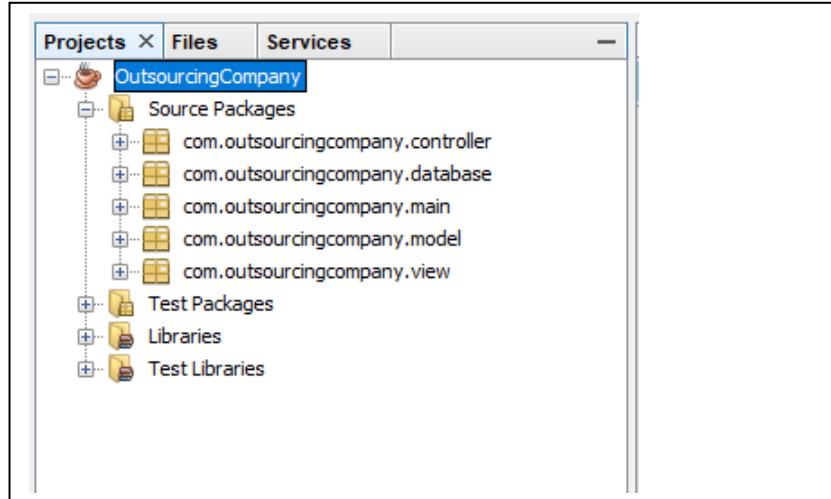
Langkah Kedua

Membuat package untuk model MVC pada Java.

Query

Masukkan query disini (Consolas 12 pt)

Screenshot



Analisa :

(Ketik hasil analisa disini)



Laporan Sementara

Langkah Ketiga

Membuat class **Employee.java** pada Package: **com.outsourcingcompany.model**.

Query

```
/*
 * To change this license header, choose
License Headers in Project Properties.
 * To change this template file, choose Tools
| Templates
 * and open the template in the editor.
 */
package com.outsourcingcompany.model;

/**
 *
 * @author Achmad Muchlasin
 */
public class Employee {
    private Integer empId;
    private String empName;
    private String empAddr;
    private String empGender;
    private String empPhoneNo;
    private String empEmail;

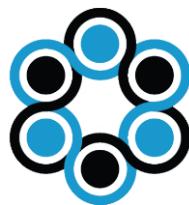
    public Integer getEmpId() {
        return empId;
    }

    public void setEmpId(Integer empId) {
        this.empId = empId;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    public String getEmpAddr() {
        return empAddr;
    }
}
```



Laporan Sementara

```
public void setEmpAddr(String empAddr) {
    this.empAddr = empAddr;
}

public String getEmpGender() {
    return empGender;
}

public void setEmpGender(String empGender) {
    this.empGender = empGender;
}

public String getEmpPhoneNo() {
    return empPhoneNo;
}

public void setEmpPhoneNo(String empPhoneNo) {
    this.empPhoneNo = empPhoneNo;
}

public String getEmpEmail() {
    return empEmail;
}

public void setEmpEmail(String empEmail)
{
    this.empEmail = empEmail;
}

}
```

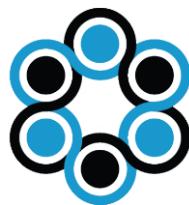


Screenshot

```
Source File: [File] [New] [Open] [Save] [Save All] [Print] [Run] [Stop] [Break] [Help] [Exit]
12 public class Employee {
13     private Integer empId;
14     private String empName;
15     private String empAddress;
16     private String empGender;
17     private String empPhoneNo;
18     private String empEmail;
19
20     public Integer getEmpId() {
21         return empId;
22     }
23
24     public void setEmpId(Integer empId) {
25         this.empId = empId;
26     }
27
28     public String getEmpName() {
29         return empName;
30     }
31
32     public void setEmpName(String empName) {
33         this.empName = empName;
34     }
35
36     public String getEmpAddress() {
37         return empAddress;
38     }
39
40     public void setEmpAddress(String empAddress) {
41         this.empAddress = empAddress;
42     }
43
44     public String getEmpGender() {
45         return empGender;
46     }
47
48     public void setEmpGender(String empGender) {
49         this.empGender = empGender;
50     }
51
52     public String getEmpPhoneNo() {
53         return empPhoneNo;
54     }
55
56     public void setEmpPhoneNo(String empPhoneNo) {
57         this.empPhoneNo = empPhoneNo;
58     }
59
60     public String getEmpEmail() {
```

Analisa :

(Ketik hasil analisa disini)



Laporan Sementara

Langkah Keempat

Membuat class **Client.java** pada Package: **com.outsourcingcompany.model**.

Query

```
/*
 * To change this license header, choose
 License Headers in Project Properties.
 * To change this template file, choose Tools
 | Templates
 * and open the template in the editor.
 */
package com.outsourcingcompany.model;

/**
 *
 * @author Achmad Muchlasin
 */
public class Client {
    private Integer clId;
    private String clName;
    private String clAddr;
    private String clEmail;
    private String clPhoneNo;

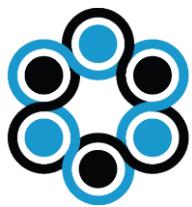
    public Integer getClId() {
        return clId;
    }

    public void setClId(Integer clId) {
        this.clId = clId;
    }

    public String getClName() {
        return clName;
    }

    public void setClName(String clName) {
        this.clName = clName;
    }

    public String getClAddr() {
        return clAddr;
    }
}
```



Laporan Sementara

```
public void setClAddr(String clAddr) {  
    this.clAddr = clAddr;  
}  
  
public String getClEmail() {  
    return clEmail;  
}  
  
public void setClEmail(String clEmail) {  
    this.clEmail = clEmail;  
}  
  
public String getClPhoneNo() {  
    return clPhoneNo;  
}  
  
public void setClPhoneNo(String  
clPhoneNo) {  
    this.clPhoneNo = clPhoneNo;  
}  
}
```

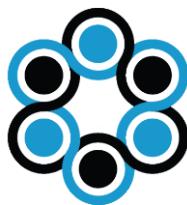
Screenshot

```
Source History |                         
```

```
11 package com.imooc;
12
13 public class Client {
14     private Integer cId;
15     private String cName;
16     private String cAddress;
17     private String cEmail;
18     private String cPhoneNo;
19
20     public Integer getCId() {
21         return cId;
22     }
23
24     public void setCId(Integer cId) {
25         this.cId = cId;
26     }
27
28     public String getcName() {
29         return cName;
30     }
31
32     public void setcName(String cName) {
33         this.cName = cName;
34     }
35
36     public String getcAddress() {
37         return cAddress;
38     }
39
40     public void setcAddress(String cAddress) {
41         this.cAddress = cAddress;
42     }
43
44     public String getcEmail() {
45         return cEmail;
46     }
47
48     public void setcEmail(String cEmail) {
49         this.cEmail = cEmail;
50     }
51
52     public String getcPhoneNo() {
53         return cPhoneNo;
54     }
55
56     public void setcPhoneNo(String cPhoneNo) {
57         this.cPhoneNo = cPhoneNo;
58     }
59
60 }
```

Analisa :

(Ketik hasil analisa disini)



Laporan Sementara

Langkah Kelima

Membuat class **Job.java** pada Package: **com.outsourcingcompany.model**.

Query

```
/*
 * To change this license header, choose
 License Headers in Project Properties.
 * To change this template file, choose Tools
 | Templates
 * and open the template in the editor.
 */
package com.outsourcingcompany.model;

/**
 *
 * @author Achmad Muchlasin
 */
public class Job {
    private Integer jobId;
    private Client client;
    private String jobName;
    private String jobCat;
    private String jobDesc;

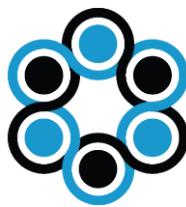
    public Integer getJobId() {
        return jobId;
    }

    public void setJobId(Integer jobId) {
        this.jobId = jobId;
    }

    public Client getClient() {
        return client;
    }

    public void setClient(Client client) {
        this.client = client;
    }

    public String getJobName() {
        return jobName;
    }
}
```



Laporan Sementara

```
public void setJobName(String jobName) {
    this.jobName = jobName;
}

public String getJobCat() {
    return jobCat;
}

public void setJobCat(String jobCat) {
    this.jobCat = jobCat;
}

public String getJobDesc() {
    return jobDesc;
}

public void setJobDesc(String jobDesc) {
    this.jobDesc = jobDesc;
}

}
```

Screenshot

```
Source File: Main.java - Line: 11 - 89
```

```
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
```

```
public class Job {
    private Integer jobId;
    private Client client;
    private String jobName;
    private String jobCat;
    private String jobDesc;

    public Integer getJobId() {
        return jobId;
    }

    public void setJobId(Integer jobId) {
        this.jobId = jobId;
    }

    public Client getClient() {
        return client;
    }

    public void setClient(Client client) {
        this.client = client;
    }

    public String getJobName() {
        return jobName;
    }

    public void setJobName(String jobName) {
        this.jobName = jobName;
    }

    public String getJobCat() {
        return jobCat;
    }

    public void setJobCat(String jobCat) {
        this.jobCat = jobCat;
    }

    public String getJobDesc() {
        return jobDesc;
    }

    public void setJobDesc(String jobDesc) {
        this.jobDesc = jobDesc;
    }
}
```

Analisa :

(Ketik hasil analisa disini)



Laporan Sementara

Langkah Keenam

Membuat class **Project.java** pada Package: **com.outsourcingcompany.model**.

Query

```
/*
 * To change this license header, choose
 License Headers in Project Properties.
 * To change this template file, choose Tools
 | Templates
 * and open the template in the editor.
 */
package com.outsourcingcompany.model;

import java.util.Date;

/**
 *
 * @author Achmad Muchlasin
 */
public class Project {
    private Integer prjId;
    private Job job;
    private Employee employee;
    private Date startDate;
    private Date dueDate;

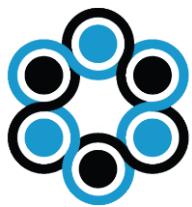
    public Integer getPrjId() {
        return prjId;
    }

    public void setPrjId(Integer prjId) {
        this.prjId = prjId;
    }

    public Job getJob() {
        return job;
    }

    public void setJob(Job job) {
        this.job = job;
    }

    public Employee getEmployee() {
        return employee;
    }
}
```



Laporan Sementara

```
public void setEmployee(Employee employee) {
    this.employee = employee;
}

public Date getStartDate() {
    return startDate;
}

public void setStartDate(Date startDate)
{
    this.startDate = startDate;
}

public Date getDueDate() {
    return dueDate;
}

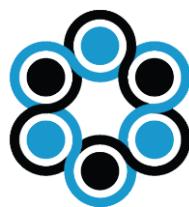
public void setDueDate(Date dueDate) {
    this.dueDate = dueDate;
}

}
```

Screenshot

Analisa :

(Ketik hasil analisa disini)

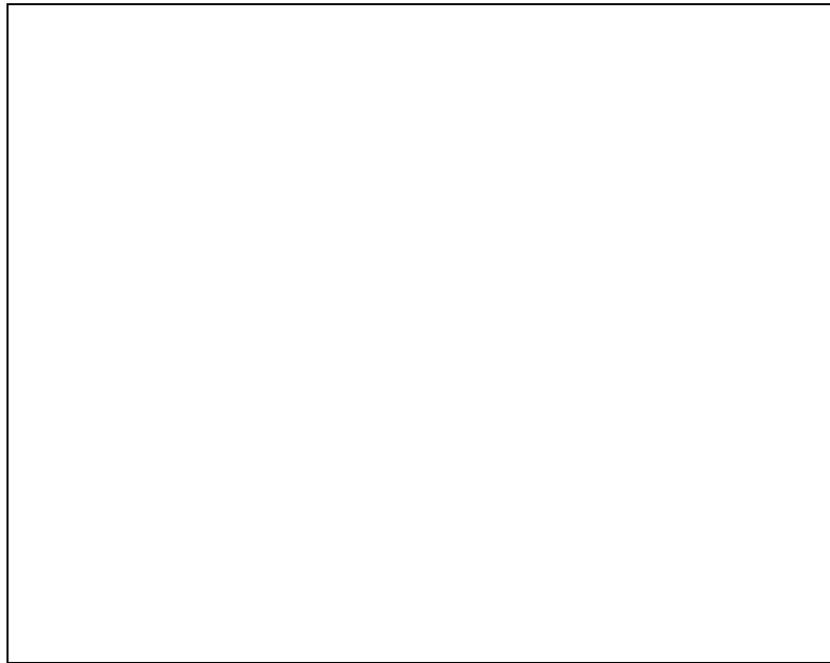


Laporan Sementara

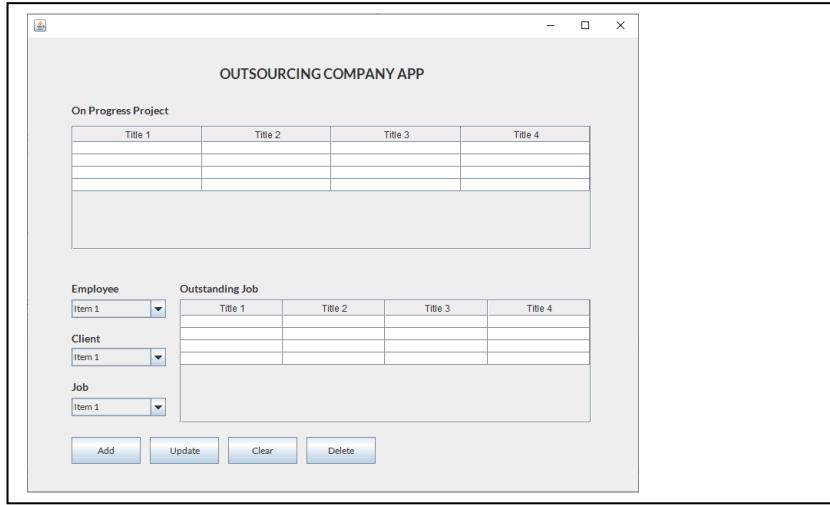
Langkah Ketujuh

Membuat class **ProjectView.java** pada Package: **com.outsourcingcompany.view**

Query



Screenshot



Analisa :

(Ketik hasil analisa disini)



KRITIK & SARAN

FAKULTAS TEKNIK ELEKTRO DAN
TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA
SURABAYA
2020



Kritik & Saran

Laboratorium

Basis Data

Kritik :

- + Fasilitas seperti komputer / PC serta meja, kursi, dan ruangannya sudah cukup nyaman. Hanya saja kurang perawatan.
- + Pengkabelan atau keringinan penyetelan kabel di dalam ruangan masih terlalu banyak. Praktikan sering tersandung kabel duri.

Saran :

- + Overall sudah memenuhi kebutuhan dan kenyamanan praktikan, namun perlu maintenance atau perawatan yang setiap hari rutin agar ruangan lebih nyaman lagi.
- + Pada sistem paralel saat ini, sistem Laboratorium yang disediakan untuk memenuhi praktikum praktikan sudah cukup bagi dan tidak mempersulit praktikan dalam melaksanakan praktikum. Semoga semakin lancar ke depannya .



Kritik & Saran

Asistem

Laberatorium Basis Data

+) March Angga

Kritik : - Sudah cukup fast respond terhadap komunitas dengan praktisan, namun untuk waktu asisten masih kurang.

Saran : - Sudah baik mas, dipertahankan yang sekarang namun ditambah lagi untuk waktu asistennya.

+) Khisby Al Ghafari

Kritik : Belum ada kritik, sejauh ini sudah cukup laius.

Saran : Pertahankan ya mas, semoga selalu sebar dengan praktisan.

+) Kiegullah Nadhif Zahirky

Kritik : Belum ada kritik, sejauh ini sudah cukup laius.

Saran : Pertahankan ya mas, semoga selalu sebar dengan praktisan.

+) Uta Shania B.

Kritik : Belum ada kritik, sejauh ini sudah cukup laius dan yang paling ramah ke praktisan.

Saran : Pertahankan ya mbak, semoga selalu sebar dengan praktisan.

+) Hilmy Maulana Ilmy

Kritik : Belum ada kritik, sejauh ini sudah cukup laius.

Saran : Pertahankan ya mbak, semoga selalu sebar dengan praktisan.

+) Aldij Wachid Arifin

Kritik : Belum ada kritik, sejauh ini sudah cukup laius.

Saran : Pertahankan ya mas, semoga selalu sebar dengan



Kritik & Saran

1) Taqiyuddin Nursy

Kritik : Belum ada kritik, sejauh ini sudah cukup bagus

Saran : Perbaikan juga mas, senyaja selaku sasaran
dengan tingkah lelu don sikap praktikam.

<https://github.com/muchlaseen/project-ta-basdat>