15001055 OhSeok Shin

1. Project title and a brief description of the project

**Apartment company record**

Apartment rental company has information of their residents, managers, buildings, apartments, contracts, payment history and etc. to arrange the information of their customers and assets of the company.

1. “Drop table,” “Create table,” and “Insert Into ..” commands

--Drop Tables:

drop table Contract;

drop table Facility;

drop table Damaged;

drop table Package;

drop table Rent\_pay;

drop table complain;

drop table Departments;

drop table Apartment;

drop table Building;

drop table Resident;

drop table Manager;

drop table Promotion;

CREATE TABLE Building

(Building# INT NOT NULL ,

Dep VARCHAR(15) NOT NULL ,

Room\_Count INT ,

Rate INT ,

Building\_Size VARCHAR(7) ,

Building\_Name VARCHAR2(64) ,

PRIMARY KEY (Building#)

) ;

Insert into BUILDING values (1,'Building15',2,550,'132yd','Alpha');

Insert into BUILDING values (2,'Building234',3,500,'156yd','Beta');

Insert into BUILDING values (3,'Building234',3,500,'156yd','Gamma');

Insert into BUILDING values (4,'Building234',3,500,'156yd','Delta');

Insert into BUILDING values (5,'Building15',4,450,'100yd','Epsilon');

Insert into BUILDING values (6,'Building67',4,450,'100yd','Zeta');

Insert into BUILDING values (7,'Building67',2,550,'135yd','Eta');

Insert into BUILDING values (8,'ClubHouse',null,null,'100yd','Oxford ClubHouse');

Insert into BUILDING values (9,'Cargo',null,null,'30yd','Cargo');

Insert into BUILDING values (10,'Janitor\_room',null,null,'35yd','Janitor room');

Commit;

CREATE TABLE Apartment

(apt# INT NOT NULL ,

Building# INT NOT NULL ,

Pet\_allowence CHAR(1),

Ele\_Bill decimal(5,2),

Res\_count INT ,

PRIMARY KEY (apt#),

FOREIGN KEY (Building#) REFERENCES Building(Building#)) ;

Insert into Apartment values('1101', '1' ,'o' , '40', '1');

Insert into Apartment values('1102', '1' ,'o' , '35', '1');

Insert into Apartment values('2101', '2' ,'o' , '80', '3');

Insert into Apartment values('2102', '2' ,'o' , '', '');

Insert into Apartment values('3101', '3' ,'o' , '', '');

Insert into Apartment values('3102', '3' ,'x' , '55', '2');

Insert into Apartment values('4101', '4' ,'x' , '', '');

Insert into Apartment values('4201', '4' ,'x' , '40', '1');

Insert into Apartment values('5101', '5' ,'x' , '', '');

Insert into Apartment values('5201', '5' ,'x' , '', '');

Insert into Apartment values('6101', '6' ,'o' , '35', '1');

Insert into Apartment values('6201', '6' ,'o' , '', '');

Insert into Apartment values('7101', '7' ,'o' , '40', '1');

Insert into Apartment values('7201', '7' ,'x' , '', '');

Commit;

CREATE TABLE Resident

(ResID INT NOT NULL PRIMARY KEY,

Res\_Fname Char(12) Not null,

Res\_Lname Char(12) Not null,

Res\_phone varchar(15),

Res\_email varchar(25));

INSERT INTO Resident Values('111111', 'Krag', 'Dawn', '662-111-1111','qqq@google.com');

INSERT INTO Resident Values('222222', 'Tim', 'Cook', '662-222-2222','www@google.com');

INSERT INTO Resident Values('333333', 'Kenny', 'Tong', '662-333-3333','eee@google.com');

INSERT INTO Resident Values('444444', 'Sandie', 'Gardner', '662-444-4444','rrr@google.com');

INSERT INTO Resident Values('555555', 'Kim', 'Baker', '662-555-5555','ttt@google.com');

INSERT INTO Resident Values('666666', 'Lewes', 'Kedi', '662-666-6666','yyy@google.com');

INSERT INTO Resident Values('777777', 'George', 'Hal', '662-777-7777','aaa@google.com');

INSERT INTO Resident Values('888888', 'Marie', 'Smith', '662-888-8888','sss@google.com');

INSERT INTO Resident Values('999999', 'Simon', 'Dominic', '662-999-9999','qwe@google.com');

INSERT INTO Resident Values('101010', 'Carmen', 'Monahun', '662-101-1010','asd@google.com');

commit;

CREATE TABLE Manager

(MgrID char(7) Not Null Primary Key,

Mgr\_Fname Char(12) Not Null,

Mgr\_Lname Char(12) Not Null,

Mgr\_Phone Varchar(15),

Mgr\_email Varchar(30),

Mgr\_password Varchar(25)

);

INSERT INTO Manager values('M123123', 'Jim', 'Bon', '662-123-1234', 'ddd@google.com', ’ddd’);

INSERT INTO Manager values('M321321', 'Susan', 'Merc', '662-321-3210', 'fff@google.com', ’fff’);

INSERT INTO Manager values('M456456', 'Tom', 'Kale', '662-456-4567', 'ggg@google.com', ‘ggg’);

INSERT INTO Manager values('M654654', 'Anna', 'Galler', '662-654-6543', 'zzz@google.com', ‘zzz’);

INSERT INTO Manager values('M789789', 'Paul', 'King', '662-789-7890', 'xxx@google.com', ‘xxx’);

INSERT INTO Manager values('M987987', 'Lee', 'Timber', '662-987-9876', 'ccc@google.com', ‘ccc’);

INSERT INTO Manager values('M134679', 'Kevin', 'Nell', '662-134-4679', 'zxc@google.com', ‘zxc’);

INSERT INTO Manager values('M467913', 'Joe', 'Col', '662-467-7913', 'poi@google.com', ‘poi’);

INSERT INTO Manager values('M791346', 'Anderson', 'Carl', '662-791-1346', 'lkj@google.com', ‘lkj’);

commit;

CREATE TABLE Promotion

(Promo# char(7) NOT NULL,

Deduction DECIMAL (5,2),

Promo\_Exp CHAR(8),

Promo\_name VARCHAR(18),

Bonus DECIMAL(5, 2),

PRIMARY KEY(Promo#)

);

INSERT INTO Promotion values('pro1', 0.25,'6M', 'Short\_term\_renewal', '');

INSERT INTO Promotion values('pro2', 0.05, '6M', 'Short\_term\_extend', 50.00);

INSERT INTO Promotion values('pro3', 0.08, '1Y', 'long\_term\_renewal', 50.00);

INSERT INTO Promotion values('pro4', 0.1, '1Y', 'Long\_term\_extend', 100.00);

INSERT INTO Promotion values('pro5', 0.15, '2Y', '2yr\_ctr', 200.00);

INSERT INTO Promotion values('pro6', 0.15, '1Y', '1yr\_special', 100.00);

INSERT INTO Promotion values('pro7', 0.20, '2Y', '2yr\_special', 200.00);

INSERT INTO Promotion values('pro8', 0, '', 'Welcome\_gift\_50', 50.00);

INSERT INTO Promotion values('pro9', 0, '', 'Welcome\_gift\_100', 100.00);

INSERT INTO Promotion values('pro10', 0.25, '2Y', 'Special\_pro', 150.00);

commit;

CREATE TABLE Contract

( ResID INT NOT NULL ,

apt# INT NOT NULL ,

Con\_start\_Date DATE,

Con\_Exp\_Date DATE,

Promo# char(7) REFERENCES Promotion(Promo#),

PRIMARY KEY (ResId),

FOREIGN KEY(apt#) REFERENCES Apartment(apt#)

);

INSERT INTO Contract values('111111', '1101', to\_date('08/01/2017', 'mm/dd/yyyy'), to\_date('01/31/2018', 'mm/dd/yyyy'), 'pro2');

INSERT INTO Contract values('222222', '1102', to\_date('01/01/2017', 'mm/dd/yyyy'), to\_date('12/31/2017', 'mm/dd/yyyy'), 'pro3');

INSERT INTO Contract values('333333', '2101', to\_date('08/01/2017', 'mm/dd/yyyy'), to\_date('07/31/2018', 'mm/dd/yyyy'), 'pro4');

INSERT INTO Contract values('444444', '2101', to\_date('08/01/2017', 'mm/dd/yyyy'), to\_date('07/31/2018', 'mm/dd/yyyy'), 'pro3');

INSERT INTO Contract values('555555', '2101', to\_date('07/01/2017', 'mm/dd/yyyy'), to\_date('12/31/2018', 'mm/dd/yyyy'), 'pro1');

INSERT INTO Contract values('666666', '3102', to\_date('08/01/2017', 'mm/dd/yyyy'), to\_date('07/31/2018', 'mm/dd/yyyy'), 'pro4');

INSERT INTO Contract values('777777', '3102', to\_date('08/01/2017', 'mm/dd/yyyy'), to\_date('07/31/2018', 'mm/dd/yyyy'), 'pro4');

INSERT INTO Contract values('888888', '4201', to\_date('01/01/2017', 'mm/dd/yyyy'), to\_date('12/31/2018', 'mm/dd/yyyy'), 'pro5');

INSERT INTO Contract values('999999', '6101', to\_date('01/01/2016', 'mm/dd/yyyy'), to\_date('12/31/2017', 'mm/dd/yyyy'), 'pro7');

INSERT INTO Contract values('101010', '7101', to\_date('08/01/2017', 'mm/dd/yyyy'), to\_date('07/31/2018', 'mm/dd/yyyy'), 'pro9');

commit;

CREATE TABLE Complain

(Com# CHAR(4) NOT NULL ,

ResID INT,

Issue VarChar(20),

Com\_rec\_date DATE,

Com\_solved\_date DATE,

PRIMARY KEY (Com#, ResID)

);

Insert into Complain values('C1', '333333', 'Sink', to\_date('08/27/2017', 'mm/dd/yyyy'), to\_date('08/30/2017', 'mm/dd/yyyy'));

Insert into Complain values('C2', '333333', 'Bathtub', to\_date('08/27/2017', 'mm/dd/yyyy'), to\_date('08/30/2017', 'mm/dd/yyyy'));

Insert into Complain values('C3', '111111', 'Stove', to\_date('09/02/2017', 'mm/dd/yyyy'), to\_date('09/08/2017', 'mm/dd/yyyy'));

Insert into Complain values('C4', '777777', 'LightBulb', to\_date('09/11/2017', 'mm/dd/yyyy'), to\_date('09/13/2017', 'mm/dd/yyyy'));

Insert into Complain values('C5', '777777', 'Electric', to\_date('09/11/2017', 'mm/dd/yyyy'), to\_date('09/13/2017', 'mm/dd/yyyy'));

Insert into Complain values('C6', '666666', 'Disposal', to\_date('09/24/2017', 'mm/dd/yyyy'), to\_date('09/29/2017', 'mm/dd/yyyy'));

Insert into Complain values('C7', '666666', 'Dish\_washer', to\_date('09/24/2017', 'mm/dd/yyyy'), to\_date('09/26/2017', 'mm/dd/yyyy'));

Insert into Complain values('C8', '222222', 'Internet', to\_date('10/12/2017', 'mm/dd/yyyy'), to\_date('10/18/2017', 'mm/dd/yyyy'));

Insert into Complain values('C9', '111111', 'Sink', to\_date('10/20/2017', 'mm/dd/yyyy'), to\_date('10/22/2017', 'mm/dd/yyyy'));

Insert into Complain values('C10', '888888', 'Electric', to\_date('11/08/2017', 'mm/dd/yyyy'), to\_date('', 'mm/dd/yyyy'));

commit;

CREATE TABLE Departments

(Dep VARCHAR(15) NOT NULL PRIMARY KEY,

MgrID Char(7) REFERENCES Manager(MgrID),

Hotline\_num varchar(3),

Location Varchar(20),

Opening\_hour Char(15)

);

INSERT INTO Departments values('Frontdesk', 'M456456', '101', 'ClubHouse', '07:30~17:00');

INSERT INTO Departments values('Office', 'M654654', '102', 'ClubHouse', '07:30~17:00');

INSERT INTO Departments values('Storage', ‘M789789’, '', 'ClubHouse', 'permanent');

INSERT INTO Departments values('Guestroom', 'M456456', '', 'ClubHouse', 'permanent');

INSERT INTO Departments values('Mailbox', ‘M789789’, '', 'ClubHouse', 'permanent');

INSERT INTO Departments values('Building15', 'M123123', '103', 'Apartment', '08:00~16:00');

INSERT INTO Departments values('Building234', 'M321321', '104', 'Apartment', '08:00~16:00');

INSERT INTO Departments values('Building67', 'M987987', '105', 'Apartment', '08:00~16:00');

INSERT INTO Departments values('Janitor', 'M134679', '106', 'FrontYard', '09:00~18:00');

INSERT INTO Departments values('Headquarter', 'M791346', '100', 'Oxford', '08:00~17:00');

commit;

CREATE TABLE Facility

( Facil# Char(3) NOT NULL PRIMARY KEY,

Dep VARCHAR(15),

Item Char(25),

Fac\_cost decimal(6, 2),

Acquision\_date DATE,

FOREIGN KEY(Dep) REFERENCES Departments(Dep)

);

Insert Into Facility Values('F1', 'Office', 'Desktop', 700, to\_date('07/25/2015', 'mm/dd/yyyy'));

Insert Into Facility Values('F2', 'Office', 'Printer', 350, to\_date('07/25/2015', 'mm/dd/yyyy'));

Insert Into Facility Values('F3', 'Office', 'Fax', 200, to\_date('07/25/2015', 'mm/dd/yyyy'));

Insert Into Facility Values('F4', 'Frontdesk', 'Laptop', 750, to\_date('07/25/2015', 'mm/dd/yyyy'));

Insert Into Facility Values('F5', 'Frontdesk', 'Coffe\_machine', 325, to\_date('03/13/2016', 'mm/dd/yyyy'));

Insert Into Facility Values('F6', 'Guestroom', 'Television', 450, to\_date('03/13/2016', 'mm/dd/yyyy'));

Insert Into Facility Values('F7', 'Guestroom', 'Projector', 250, to\_date('03/13/2016', 'mm/dd/yyyy'));

Insert Into Facility Values('F8', 'Guestroom', 'Vending\_machine', 650, to\_date('07/05/2016', 'mm/dd/yyyy'));

Insert Into Facility Values('F9', 'Office', 'Carpet', 200, to\_date('08/10/2016', 'mm/dd/yyyy'));

Insert Into Facility Values('F10', 'Janitor', 'Heater', 150, to\_date('08/16/2016', 'mm/dd/yyyy'));

commit;

CREATE TABLE Damaged

(DamID char(4) NOT NULL Primary Key,

apt# INT ,

Dam\_Item varchar(20),

Dam\_Cost decimal(6, 2),

Fixed char,

foreign key(apt#) references Apartment(apt#)

);

Insert into Damaged values('D1', '1102', 'Window', 100, 'o');

Insert into Damaged values('D2', '2101', 'Refregerator', 250, 'o');

Insert into Damaged values('D3', '2102', 'Door', 80, 'o');

Insert into Damaged values('D4', '5101', 'Sink', 30, 'o');

Insert into Damaged values('D5', '4101', 'Pipe', 50, 'x');

Insert into Damaged values('D6', '4201', 'Pipe', 50, 'x');

Insert into Damaged values('D7', '6101', 'Wall', 40, 'x');

Insert into Damaged values('D8', '7201', 'Couch', 35, 'o');

Insert into Damaged values('D9', '3102', 'Window', 100, 'x');

Insert into Damaged values('D10', '1101', 'Dishwasher', 125, 'x');

commit;

CREATE TABLE Package

(Pac# char(6) not null Primary key,

ResID INT NOT NULL,

Pac\_Size char,

Pac\_arr DATE,

Pickup char,

FOREIGN KEY(ResID) references Resident(ResID)

);

INSERT INTO Package values('pac1', '222222','M', To\_Date('08/09/2017', 'mm/dd/yyyy'), 'o');

INSERT INTO Package values('pac2', '222222','L', To\_Date('09/10/2017', 'mm/dd/yyyy'), 'x');

INSERT INTO Package values('pac3', '555555','L', To\_Date('09/12/2017', 'mm/dd/yyyy'), 'o');

INSERT INTO Package values('pac4', '777777','S', To\_Date('09/12/2017', 'mm/dd/yyyy'), 'o');

INSERT INTO Package values('pac5', '888888','M', To\_Date('09/17/2017', 'mm/dd/yyyy'), 'o');

INSERT INTO Package values('pac6', '111111','M', To\_Date('09/22/2017', 'mm/dd/yyyy'), 'o');

INSERT INTO Package values('pac7', '111111','L', To\_Date('09/22/2017', 'mm/dd/yyyy'), 'x');

INSERT INTO Package values('pac8', '999999','S', To\_Date('10/29/2017', 'mm/dd/yyyy'), 'x');

INSERT INTO Package values('pac9', '444444','L', To\_Date('11/03/2017', 'mm/dd/yyyy'), 'o');

INSERT INTO Package values('pac10', '222222','M', To\_Date('11/04/2017', 'mm/dd/yyyy'), 'x');

commit;

CREATE TABLE Rent\_pay

(Pay# Char(6) not null primary key,

ResID INT not null,

paid\_date date,

Paid\_amt decimal (6, 2),

Foreign key (ResID) references Resident (ResID)

);

Insert into Rent\_pay values('pay001', '222222', to\_date('10/01/2017', 'mm/dd/yyyy'), 585);

Insert into Rent\_pay values('pay002', '444444', to\_date('10/02/2017', 'mm/dd/yyyy'), 527);

Insert into Rent\_pay values('pay003', '777777', to\_date('10/02/2017', 'mm/dd/yyyy'), 527);

Insert into Rent\_pay values('pay004', '222222', to\_date('11/01/2017', 'mm/dd/yyyy'), 535);

Insert into Rent\_pay values('pay005', '333333', to\_date('11/02/2017', 'mm/dd/yyyy'), 527);

Insert into Rent\_pay values('pay006', '555555', to\_date('11/02/2017', 'mm/dd/yyyy'), 527);

Insert into Rent\_pay values('pay007', '666666', to\_date('11/02/2017', 'mm/dd/yyyy'), 527);

Insert into Rent\_pay values('pay008', '777777', to\_date('11/03/2017', 'mm/dd/yyyy'), 527);

Insert into Rent\_pay values('pay009', '888888', to\_date('11/03/2017', 'mm/dd/yyyy'), 540);

Insert into Rent\_pay values('pay010', '111111', to\_date('11/04/2017', 'mm/dd/yyyy'), 590);

commit;

1)

--provide apartmenet information using package's function and procedure

--Building number, building name, building manager and assigned department,

--people who lives in the apartment, apartment monthly rate, and building size

--create package head of sub\_info\_name.

--Exception string shows, in case of wrong apartment info input.

------------------------------

--create package

create or replace PACKAGE sub\_info\_name

IS

FUNCTION sub\_info\_mgr\_pos

(mgr\_id MANAGER.MGRID%type)

RETURN VARCHAR2;

procedure sub\_info\_damaged

(aptnum Damaged.apt#%type);

procedure sub\_info\_pet

(p\_allw Apartment.Pet\_allowence%type);

End sub\_info\_name;

/

---------------------------------------------------

--package body sub information of apartment and building

create or replace PACKAGE BODY sub\_info\_name

IS

-----------------------------

-- Damaged Item finder

procedure sub\_info\_damaged

(aptnum IN Damaged.apt#%type)

IS

DAM\_num INT;

BEGIN

SELECT COUNT(damID)

INTO DAM\_num

FROM damaged

WHERE apt# = aptnum;

IF DAM\_num > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('APT# '||aptnum|| ' has '||DAM\_num||' Damaged Item(s).');

ELSE

DBMS\_OUTPUT.PUT\_LINE('APT#' ||aptnum|| ' is just fine like new!');

END IF;

END sub\_info\_damaged;

----------------------------------------------------

--INFORMATION OF MANAGER'S POSITION

FUNCTION sub\_info\_mgr\_pos

(mgr\_id MANAGER.MGRID%type)

RETURN VARCHAR2

IS

ret\_mgr\_pos VARCHAR2(30);

BEGIN

IF mgr\_id = 'M123123' THEN

ret\_mgr\_pos := 'Building\_15\_Manager';

Elsif mgr\_id = 'M321321' THEN

ret\_mgr\_pos := 'Building\_234\_Manager';

Elsif mgr\_id = 'M456456' THEN

ret\_mgr\_pos := 'Front desk';

Elsif mgr\_id = 'M654654' THEN

ret\_mgr\_pos := 'Office Manager';

Elsif mgr\_id = 'M789789' THEN

ret\_mgr\_pos := 'Property Manager';

Elsif mgr\_id = 'M987987' THEN

ret\_mgr\_pos := 'Building\_78\_Manager';

Elsif mgr\_id = 'M134679' THEN

ret\_mgr\_pos := 'Janitor';

Elsif mgr\_id = 'M467913' THEN

ret\_mgr\_pos := 'Janitor';

Elsif mgr\_id = 'M791346' THEN

ret\_mgr\_pos := 'Headquarter';

ELSE

ret\_mgr\_pos := 'N/A';

END IF;

RETURN ret\_mgr\_pos;

END sub\_info\_mgr\_pos;

-------------------------------

--pet allowence finder

PROCEDURE sub\_info\_pet

(p\_allw Apartment.Pet\_allowence%type)

IS

BEGIN

IF p\_allw = 'o' THEN

DBMS\_OUTPUT.PUT\_LINE('Pet allowence : Yes');

ELSE

DBMS\_OUTPUT.PUT\_LINE('You cannot bring your pet. sorry');

END IF;

END sub\_info\_pet;

END;

/

------------------------------

--run main sql

SET serveroutput ON;

DECLARE

b\_number BUILDING.BUILDING#%type := '';

depart BUILDING.Dep%type := '';

apt\_num Apartment.apt#%type;

rooms number;

b\_rate decimal(6,2);

b\_size Building.Building\_Size%type;

b\_name varchar2(64);

pet\_allw Apartment.Pet\_allowence%type;

num\_res number;

b\_mgr\_fn Manager.Mgr\_Fname%type;

b\_mgr\_ln Manager.Mgr\_Lname%type;

b\_mgr\_id Manager.MgrID%type;

b\_mgr\_ph Manager.Mgr\_Phone%type;

b\_mgr\_pos VARCHAR2(30);

ex\_apt EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_apt,-06502);

BEGIN

apt\_num := '&ENTER\_APARTMENT\_NUMBER'; --enter 1101

SELECT Building#, Pet\_allowence, Res\_count

INTO b\_number, pet\_allw, num\_res FROM Apartment

WHERE apt# = apt\_num;

SELECT Dep, Room\_Count, Rate, Building\_Size, Building\_name

INTO depart, rooms, b\_rate, b\_size, b\_name

FROM Building

WHERE Building# = b\_number;

SELECT M.MgrID, Mgr\_Fname, Mgr\_Lname, Mgr\_Phone

INTO b\_mgr\_id, b\_mgr\_fn, b\_mgr\_ln, b\_mgr\_ph

FROM Manager M, Departments D, Building B, Apartment A

WHERE M.MgrID = D.MgrID

AND D.Dep = B.Dep

AND B.Building# = A.Building#

AND A.apt# = apt\_num;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Thanks for using apartment info finder');

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Information of Apartment number :'||apt\_num);

DBMS\_OUTPUT.PUT\_LINE('It is on Building number :'||b\_number);

DBMS\_OUTPUT.PUT\_LINE('The building has a name of : '||b\_name);

DBMS\_OUTPUT.PUT\_LINE('managed by Department :'||depart);

DBMS\_OUTPUT.PUT\_LINE('Building manager is : '

||trim(b\_mgr\_fn)||' '||trim(b\_mgr\_ln));

DBMS\_OUTPUT.PUT\_LINE('It has '||rooms||' rooms');

DBMS\_OUTPUT.PUT\_LINE('Now '||num\_res||' people in the unit');

DBMS\_OUTPUT.PUT\_LINE('Rate : $'||b\_rate);

DBMS\_OUTPUT.PUT\_LINE('Building Size :'||b\_size);

sub\_info\_name.sub\_info\_pet(pet\_allw);

sub\_info\_name.sub\_info\_damaged(apt\_num);

DBMS\_OUTPUT.PUT\_LINE('');

b\_mgr\_pos := sub\_info\_name.sub\_info\_mgr\_pos(b\_mgr\_id);

DBMS\_OUTPUT.PUT\_LINE('For more information, contact with :');

DBMS\_OUTPUT.PUT\_LINE(b\_mgr\_pos||' ;'||b\_mgr\_ph);

EXCEPTION

WHEN ex\_apt THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Apartment number input error!');

DBMS\_OUTPUT.PUT\_LINE('Try with proper Apartment number!');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('APT#: "'||apt\_num||'" is not in the data. ');

DBMS\_OUTPUT.PUT\_LINE('Try other Apartment number. ');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('A problem has ocurred in retrieving your saved data.');

DBMS\_OUTPUT.PUT\_LINE('Please contact with Tech Support via email.');

END;

/

--------------------------------------------------

--output check

--when 1101 entered

/\*

Thanks for using apartment info finder

Information of Apartment number :1101

It is on Building number :1

The building has a name of : Alpha

managed by Department :Building15

Building manager is : Jim Bon

It has 2 rooms

Now 1 people in the unit

Rate : $550

Building Size :132yd

Pet allowence : Yes

For more information, contact with :

Building\_15\_Manager ;662-123-1234

\*/

------------------------------------------

--when 3101 entered

/\*

Thanks for using apartment info finder

Information of Apartment number :3101

It is on Building number :3

The building has a name of : Gamma

managed by Department :Building234

Building manager is : Susan Merc

It has 3 rooms

Now people in the unit

Rate : $500

Building Size :156yd

Pet allowence : Yes

APT#3101 is just fine like new!

For more information, contact with :

Building\_234\_Manager ;662-321-3210

\*/

----------------------------

--when 8446 entered

/\*

8446 is not in the data.

Try other Apartment number.

\*/

---------------------------------

--when 'abc123' entered

/\*

Apartment number input error!

Try with proper Apartment number!

\*/

2)

--Shows Resident's information

--put one of Residents ResID then it shows their information.

--Their name, phone number, email, apt#, contract start date,

--contract expire date, promotion number,

--remained package using package counter function

-- and rant\_pay information using rent\_pay\_check procedure.

------------------------------

--create package

create or replace PACKAGE res\_info\_pkg

IS

FUNCTION res\_pack\_rema

(R\_id Package.ResID%type)

RETURN number;

FUNCTION res\_promo\_full\_info

(promo\_num promotion.promo#%type )

RETURN string;

procedure res\_rent\_pay\_hist

(R\_id Rent\_Pay.ResID%type);

End res\_info\_pkg;

/

---------------------------------------------------

--package body sub information of apartment and building

create or replace PACKAGE BODY res\_info\_pkg

AS

--package counter

FUNCTION res\_pack\_rema

(R\_id Package.ResID%type)

RETURN number

IS

pack\_count number;

BEGIN

SELECT count(\*)

INTO pack\_count

FROM Package

WHERE ResID = R\_id

AND Pickup = 'x';

RETURN pack\_count;

END res\_pack\_rema;

----------------------------------

--promotion name information

FUNCTION res\_promo\_full\_info

(promo\_num promotion.promo#%type)

RETURN string

IS

ret\_promo\_info string(200);

Dedu number(5,2);

proex char (8);

prona varchar(20);

bon number(5,2);

BEGIN

SELECT Deduction, Promo\_Exp, Promo\_name, Bonus

INTO Dedu, proex, prona, bon

FROM Promotion

WHERE Promo# = 'pro1';

ret\_promo\_info := 'Deduction: 0' ||trim(Dedu)||

' Promotion period: '||trim(proex)||

' promotion Name: '||trim(prona)||

' Bonus Gift card: $'||trim(bon);

RETURN ret\_promo\_info;

END res\_promo\_full\_info;

-------------------------------

--rent pay history form procedure

PROCEDURE res\_rent\_pay\_hist

(R\_ID Rent\_Pay.ResID%type)

IS

cursor cur\_rent is

select ResID, paid\_date, paid\_amt

from rent\_pay

where ResID = R\_ID;

BEGIN

FOR rec\_rent IN cur\_rent

LOOP

IF rec\_rent.ResID = R\_ID then

DBMS\_OUTPUT.PUT\_LINE ('Paid date: '||trim (rec\_rent.paid\_date));

DBMS\_OUTPUT.PUT\_LI NE ('Paid amount: '||trim (rec\_rent.paid\_amt));

DBMS\_OUTPUT.PUT\_LINE ('');

end if;

END LOOP;

END res\_rent\_pay\_hist;

END;

/

------------------------------

--run main sql

SET serveroutput ON;

DECLARE

res\_id RESIDENT.RESID%type;

res\_fn RESIDENT.Res\_Fname%type := '';

res\_ln RESIDENT.Res\_Lname%type := '';

res\_ph RESIDENT.RES\_PHONE%type := '';

res\_em RESIDENT.Res\_email%type := '';

R\_apt\_num Contract.apt#%type:= '';

R\_con\_st CONTRACT.CON\_START\_DATE%type:= '';

R\_con\_ex CONTRACT.CON\_EXP\_DATE%type:= '';

R\_pro\_num promotion.promo#%type:='';

res\_pac\_count number:='';

res\_promo\_info string(200):='';

ex\_resid EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_resid,-06502);

BEGIN

res\_id := '&ENTER\_RESIDENT\_ID';

SELECT Res\_Fname, Res\_Lname, Res\_phone, Res\_email

INTO res\_fn, res\_ln, res\_ph, res\_em

FROM RESIDENT

WHERE ResID = res\_id;

SELECT apt#, Con\_start\_Date, Con\_Exp\_Date, Promo#

INTO R\_apt\_num, R\_con\_st, R\_con\_ex, R\_pro\_num

FROM Contract

WHERE ResID = res\_id;

--------------------------------------------

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Thanks for using Resident finder.');

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Information of resident number :'||res\_id);

DBMS\_OUTPUT.PUT\_LINE('NAME :'||trim(res\_fn)||

' '||trim(res\_ln));

DBMS\_OUTPUT.PUT\_LINE('Resident phone number : '||res\_ph);

DBMS\_OUTPUT.PUT\_LINE('Resident email address : '||res\_em);

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('The Resident has a contract of unit : '||R\_apt\_num);

DBMS\_OUTPUT.PUT\_LINE('Contract Valid date : '

||R\_con\_st||' ~ '|| R\_con\_ex);

IF R\_pro\_num = '' THEN

DBMS\_OUTPUT.PUT\_LINE('No promotion adjusted on this Resident. ');

ELSE

res\_promo\_info := res\_info\_pkg.res\_promo\_full\_info(R\_pro\_num);

DBMS\_OUTPUT.PUT\_LINE('Resident got promotion of : '||R\_pro\_num);

DBMS\_OUTPUT.PUT\_LINE('promo info : '||res\_promo\_info);

END IF;

res\_pac\_count := res\_info\_pkg.res\_pack\_rema(res\_id);

IF res\_pac\_count > '0' then

DBMS\_OUTPUT.PUT\_LINE('The Resident has '||res\_pac\_count||' package(s).');

ELSIF

res\_pac\_count = '0' THEN

DBMS\_OUTPUT.PUT\_LINE('The Resident does not have any package.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('No data found');

END IF;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident Rent pay history : ');

res\_info\_pkg.res\_rent\_pay\_hist(res\_id);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('No Resident ID found in data ');

WHEN ex\_resid THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident ID input error!');

DBMS\_OUTPUT.PUT\_LINE('Try with proper ID');

END;

----------------------------------------------

--output check

--when '222222' entered

/\*

Thanks for using Resident finder.

Information of resident number :222222

NAME :Tim Cook

Resident phone number : 662-222-2222

Resident email address : www@google.com

The Resident has a contract of unit : 1102

Contract Valid date : 17/01/01 ~ 17/12/31

Resident got promotion of : pro3

promo info : Deduction: 0.25 Promotion period: 6M promotion Name: Short\_term\_renewal Bonus Gift card: $50

The Resident has 2 package(s).

Resident Rent pay history :

Paid date: 17/10/01

Paid amount: 585

Paid date: 17/11/01

Paid amount: 535

\*/

------------------------------------------

--when 'abc1234' entered

/\*

Resident ID input error!

Try with proper ID

\*/

------------------------------------------

3)

--Package checker

--Type one of commends that you will process.

--package\_pickup\_hist procedure for checking whether a resident has picked up his or her package or not.

--Trigger for updating package table to make a pickup history.

--when resident pick up the package then package pick up status updates 'X' to 'O'

--trigger processed and it saves data of who's, package size, package arrieved date

--and when does it picked up.

--User can check specific resident's package pick history

--by typing their resident ID.

--the other procedure, remained package finder,

--is for checking package finder which isn't picked up yet.

--It retrieves the remained packages for a specific resident.

-----------------------------------------

--create/drop dumy table

DROP TABLE Package3;

CREATE TABLE Package3 as SELECT \* FROM PACKAGE;

-------------------------

--create/drop table package update log

drop table package\_pickup\_hist;

create table package\_pickup\_hist

(Package\_Num CHAR(6),

Resident\_ID NUMBER(38,0),

package\_size CHAR(1 BYTE),

Pac\_Arrival DATE,

pickup\_date timestamp(6)

);

----------------------------

--create a trigger for updated table

create or replace TRIGGER tri\_pack\_pick

AFTER UPDATE ON Package3

FOR EACH ROW

BEGIN

INSERT INTO PACKAGE\_PICKUP\_HIST

(Package\_Num, RESIDENT\_ID, PACKAGE\_SIZE, PAC\_ARRIVAL, PICKUP\_DATE)

VALUES

(:Old.pac#, :Old.RESID, :Old.PAC\_SIZE, :Old.pac\_arr, sysdate);

END;

/

------------------------

--trigger sample

select \* from Package3 where ResID = '222222';

--resident ID:222222 's package has been picked up.

update Package3 set pickup = 'o' where Pac# = 'pac2';

select \* from Package3 where ResID = '222222';

select \* from package\_pickup\_hist;

--resident ID:222222 's package has been picked up again.

update Package3 set pickup = 'o' where Pac# = 'pac10';

select \* from Package3 where ResID = '222222';

select \* from package\_pickup\_hist;

--other person's package has been picked up

update Package3 set pickup = 'o' where Pac# = 'pac5';

select \* from Package3 where ResID = '222222'; --nothing changed on 222222's package

select \* from package\_pickup\_hist;

/

-------------------------------------------------

--create package

create or replace PACKAGE pack\_pro\_pac

IS

procedure pro\_check\_pickup\_history

(i\_resid NUMBER);

procedure pro\_check\_pac

(i\_resid NUMBER);

End pack\_pro\_pac;

/

--------------------------------------------

--package body pack\_pro\_pac

create or replace PACKAGE BODY pack\_pro\_pac

AS

--Pick up history finder

PROCEDURE pro\_check\_pickup\_history

(i\_resid NUMBER)

is

coun\_res number;

---------------------

CURSOR cur\_pack\_his IS

SELECT \* from package\_pickup\_hist;

----------------------

BEGIN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Thanks for using package pickup history finder');

DBMS\_OUTPUT.PUT\_LINE('');

-------------------------

SELECT COUNT(\*)

INTO coun\_res

FROM package\_pickup\_hist

WHERE RESIDENT\_ID = i\_resid;

IF coun\_res = '0' then

DBMS\_OUTPUT.PUT\_LINE('Resident number: '

||i\_resid

||' had no package pickup history.');

DBMS\_OUTPUT.PUT\_LINE('Or there is no data of '||i\_resid);

------------------------

ELSE

DBMS\_OUTPUT.PUT\_LINE('Resident '||i\_resid||' has '

||coun\_res||' package pickup history');

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('PACKAGE\_NUM RESIDENT\_ID SIZE PAC\_ARRIVAL PICKUP\_DATE ');

FOR rec\_pack IN cur\_pack\_his

LOOP

IF rec\_pack.RESIDENT\_ID = i\_resid

then

DBMS\_OUTPUT.PUT\_LINE(

rec\_pack.PACKAGE\_NUM||' '

||rec\_pack.RESIDENT\_ID||' '

||rec\_pack.PACKAGE\_SIZE||' '

||rec\_pack.PAC\_ARRIVAL||' '

||rec\_pack.PICKUP\_DATE

);

end if;

end loop;

end if;

END pro\_check\_pickup\_history;

-----------------------------

--remained package finder

PROCEDURE pro\_check\_pac

(i\_resid NUMBER)

is

coun\_res number;

---------------------

CURSOR cur\_pack IS

SELECT \* from package3 where pickup = 'x';

BEGIN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Thanks for using package finder for resident');

DBMS\_OUTPUT.PUT\_LINE('');

---------------------

SELECT COUNT(\*)

INTO coun\_res

FROM package3 P3

WHERE ResID = i\_resid

and P3.pickup = 'x';

-----------------------

IF coun\_res = '0' then

DBMS\_OUTPUT.PUT\_LINE('Resident number: '

||i\_resid

||' has no package to pickup.');

DBMS\_OUTPUT.PUT\_LINE('Or there is no data of '||i\_resid);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Resident '||i\_resid||' has '

||coun\_res||' packages to pickup');

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('PACKAGE\_NUM RESIDENT\_ID SIZE PAC\_ARRIVAL PICKUP');

FOR rec\_pack IN cur\_pack

LOOP

IF rec\_pack.RESID = i\_resid

then

DBMS\_OUTPUT.PUT\_LINE(

rec\_pack.PAC#||' '

||rec\_pack.RESID||' '

||rec\_pack.PAC\_SIZE||' '

||rec\_pack.PAC\_ARR||' '

||rec\_pack.PICKUP

);

end if;

end loop;

end if;

END pro\_check\_pac;

END;

/

-----------------------------

SET serveroutput ON;

DECLARE

comm varchar2(40) := '';

F\_p\_resid NUMBER(10,0) := '';

ex\_resid EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_resid,-06502);

----------------------

BEGIN

comm := '&ENTER\_COMMEND\_`pickuphistory`\_`checkpackage`';

F\_p\_resid := '&Enter\_Resident\_ID';

--------------------

IF comm = 'pickuphistory' THEN

pack\_pro\_pac.pro\_check\_pickup\_history(F\_p\_resid);

--------------------

ELSIF comm = 'checkpackage' THEN

pack\_pro\_pac.pro\_check\_pac(F\_p\_resid);

---------------------

ELSE

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Wrong commend. try again');

END IF;

-------------------------------------

EXCEPTION

WHEN ex\_resid THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident ID input error!');

DBMS\_OUTPUT.PUT\_LINE('Try with proper ID');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('No Resident ID found in data. ');

DBMS\_OUTPUT.PUT\_LINE('Try again with other ID. ');

END;

/

------------------------------------

--output check

--when pickuphistory command choose

--and '222222'entered

/\*

Thanks for using package pickup history finder

Resident 222222 has 2 package pickup history

PACKAGE\_NUM RESIDENT\_ID SIZE PAC\_ARRIVAL PICKUP\_DATE

pac2 222222 L 17/09/10 18/04/12 02:08:40.000000

pac10 222222 M 17/11/04 18/04/12 02:08:40.000000

\*/

------------------------------------

--when checkpackage command choose

--and '222222' entered

/\*

Thanks for using package finder for resident

Resident number: 222222 has no package to pickup.

Or there is no data of 222222

\*/

-----------------------------------

--when type 1q2w3e4r on command line

/\*

Wrong commend. try again

\*/

----------------------------------

--when checkpackage command choose

--and '1q2w3e' entered

/\*

Resident ID input error!

Try with proper ID

\*/

--------------------------------

4)

--a package has table modifying procedures;

--Insert and delete resident data, in case of a new resident come for an apartment,

--or a resident is no longer our customer.

--Once user choose the work which they will do by typing commend,

--the resident needs to input his orher data on the resident table.

--when new resident information adjusted, triggerS runs to update log table

-----------------------------------

--create dummy table

DROP TABLE Resident5;

CREATE TABLE Resident5

as SELECT \* FROM Resident;

----------------------------------

--create resident count log

DROP TABLE Res\_count\_log;

CREATE TABLE Res\_count\_log

(user\_name varchar2(20),

date\_time timestamp(6),

Count\_number\_record number);

/

-----------------------------------

--create package

create or replace package pac\_mod\_new

is

procedure pro\_insert\_res

(N\_resid NUMBER,

N\_resfname CHAR,

N\_reslname CHAR,

N\_resphone VARCHAR2,

N\_resemail VARCHAR2);

procedure pro\_delete\_res

(D\_resid NUMBER);

END pac\_mod\_new;

/

---------------------

--build package body

CREATE OR REPLACE PACKAGE BODY pac\_mod\_new

AS

--procedure for insert

procedure pro\_insert\_res

(N\_resid NUMBER,

N\_resfname CHAR,

N\_reslname CHAR,

N\_resphone VARCHAR2,

N\_resemail VARCHAR2)

is

BEGIN

INSERT INTO

Resident5(

RESID, RES\_FNAME,

RES\_LNAME, RES\_PHONE,

RES\_EMAIL)

VALUES(

N\_resid, N\_resfname,

N\_reslname, N\_resphone,

N\_resemail);

END pro\_insert\_res;

-------------------------

--procedure for delete

procedure pro\_delete\_res

(D\_resid NUMBER)

is

BEGIN

DELETE FROM Resident5

WHERE RESID = D\_resid;

END pro\_delete\_res;

-------------------------

END;

/

------------------------------

Create or replace trigger tri\_res\_delete

AFTER DELETE ON Resident5

DECLARE

count\_rec NUMBER :=0;

BEGIN

Delete from Res\_count\_log;

Select count(\*)

into count\_rec

from Resident5;

Insert into Res\_count\_log(

user\_name,

date\_time ,

Count\_number\_record

)

values

(user, sysdate, count\_rec);

End;

/

-----------------------

Create or replace trigger tri\_res\_insert

AFTER INSERT ON Resident5

DECLARE

count\_rec NUMBER :=0;

BEGIN

Delete from Res\_count\_log;

Select count(\*) into count\_rec from Resident5;

Insert into Res\_count\_log(

user\_name,

date\_time ,

Count\_number\_record

)

values(user, sysdate, count\_rec);

End;

/

--------------------------------

SET serveroutput ON;

DECLARE

comm varchar2(40) := '';

i\_resid NUMBER(10) := '';

i\_resfname CHAR(12) := '';

i\_reslname CHAR(12) := '';

i\_resphone VARCHAR2(15) := '';

i\_resemail VARCHAR2(25) := '';

ex\_nullresid EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_nullresid, -01400);

ex\_resid EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_resid,-06502);

BEGIN

comm := '&Enter\_your\_work\_`insert`\_`delete`';

i\_resid := '&Enter\_New\_Res\_ID\_to\_`insert`\_or\_`delete`\_`6\_number`';

IF comm = 'insert' then

i\_resfname := '&Enter\_First\_Name';

i\_reslname := '&Enter\_Last\_Name';

i\_resphone := '&Enter\_Phone\_number';

i\_resemail := '&Enter\_email';

pac\_mod\_new.pro\_insert\_res(

i\_resid,

i\_resfname,

i\_reslname,

i\_resphone,

i\_resemail);

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident data('||

i\_resid||') inserted successfully');

ELSIF comm = 'delete' THEN

pac\_mod\_new.pro\_delete\_res(i\_resid);

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident data('||

i\_resid||')deleted successfully');

ELSE

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Wrong command!');

END IF;

----------------------------

EXCEPTION

WHEN ex\_nullresid THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident ID must be entered!');

WHEN ex\_resid THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident ID input error!');

DBMS\_OUTPUT.PUT\_LINE('Try with proper ID');

END;

/

-----------------------------------

--output check

SELECT \* FROM Res\_count\_log;

SELECT \* FROM Resident5;

5)

--new resident insert manager

--user type their data of personal information and about contract.

-- insert procedure get data and insert into the dummy resident table. It automatically runs new

-- resident insert trigger, then type information of contract. It’ll saved in dummy contract table

-----------------------------------

--create dummy table

DROP TABLE Resident5;

CREATE TABLE Resident5

as SELECT \* FROM Resident;

DROP TABLE Contract5;

CREATE TABLE Contract5

as SELECT \* FROM Contract;

-----------------------------------

--create package

create or replace package pac\_ist\_new

is

procedure pro\_new\_res

(N\_resid NUMBER, N\_resfname CHAR, N\_reslname CHAR, N\_resphone VARCHAR2, N\_resemail VARCHAR2);

END pac\_ist\_new;

/

---------------------

--build package body

CREATE OR REPLACE PACKAGE BODY pac\_ist\_new

AS

--procedure for insert

procedure pro\_new\_res

(N\_resid NUMBER, N\_resfname CHAR, N\_reslname CHAR, N\_resphone VARCHAR2, N\_resemail VARCHAR2)

is

BEGIN

INSERT INTO

Resident5(RESID, RES\_FNAME, RES\_LNAME, RES\_PHONE, RES\_EMAIL)

VALUES(N\_resid, N\_resfname, N\_reslname, N\_resphone, N\_resemail);

END pro\_new\_res;

-------------------------

END;

/

--------------------------------

SET serveroutput ON;

DECLARE

i\_resid NUMBER(10) := '';

i\_resfname CHAR(12) := '';

i\_reslname CHAR(12) := '';

i\_resphone VARCHAR2(15) := '';

i\_resemail VARCHAR2(25) := '';

i\_apt# NUMBER(10) := '';

i\_con\_sdate date;

i\_con\_edate date;

i\_promo CHAR(7) := '';

ex\_nullresid EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_nullresid, -01400);

ex\_resid EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_resid,-06502);

------------------------------------------------

BEGIN

--type new resident information in order.

i\_resid := '&Enter\_New\_Res\_ID\_to\_`insert`\_`6\_number`'; --946537

i\_resfname := '&Enter\_First\_Name'; --Jim

i\_reslname := '&Enter\_Last\_Name'; --Miorrison

i\_resphone := '&Enter\_Phone\_number'; --662-452-7894

i\_resemail := '&Enter\_email'; --JJJJ@gmail.com

i\_apt# := '&Which\_Apt\_number\_do\_you\_want\_to\_go\_`4digit`'; --5101

i\_con\_sdate := '&From\_when\_the\_contract\_start'; -- 2018-05-01

i\_con\_edate := '&When\_would\_be\_the\_contract\_expired'; --2019-05-01

i\_promo := '&You\_have\_promotion\_number'; --pro5

pac\_mod\_new.pro\_insert\_res(

i\_resid, i\_resfname, i\_reslname, i\_resphone, i\_resemail);

INSERT INTO Contract5(RESID, APT#, CON\_START\_DATE, CON\_EXP\_DATE, PROMO#)

values(i\_resid, i\_apt#, i\_con\_sdate, i\_con\_edate, i\_promo);

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident data('||i\_resid||') inserted successfully');

----------------------------

EXCEPTION

WHEN ex\_nullresid THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident ID must be entered!');

WHEN ex\_resid THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('data input error!');

DBMS\_OUTPUT.PUT\_LINE('Try with proper information');

END;

/

-----------------------------------

--output check

SELECT \* FROM Resident5;

/\*

RESID RES\_FNAME RES\_LNAME RES\_PHONE RES\_EMAIL

---------- ------------ ------------ --------------- -------------------------

111111 Krag Dawn 662-111-1111 qqq@google.com

222222 Tim Cook 662-222-2222 www@google.com

333333 Kenny Tong 662-333-3333 eee@google.com

444444 Sandie Gardner 662-444-4444 rrr@google.com

555555 Kim Baker 662-555-5555 ttt@google.com

666666 Lewes Kedi 662-666-6666 yyy@google.com

777777 George Hal 662-777-7777 aaa@google.com

888888 Marie Smith 662-888-8888 sss@google.com

999999 Simon Dominic 662-999-9999 qwe@google.com

101010 Carmen Monahun 662-101-1010 asd@google.com

946537 Jim Miorrison 662-452-7894 [JJJJ@gmail.com](mailto:JJJJ@gmail.com)

\*/

----------------------------------------------------------------------------------

SELECT \* FROM Contract5;

/\*

RESID APT# CON\_STAR CON\_EXP\_ PROMO#

---------- ---------- -------- -------- -------

111111 1101 17/08/01 18/01/31 pro2

222222 1102 17/01/01 17/12/31 pro3

333333 2101 17/08/01 18/07/31 pro4

444444 2101 17/08/01 18/07/31 pro3

555555 2101 17/07/01 18/12/31 pro1

666666 3102 17/08/01 18/07/31 pro4

777777 3102 17/08/01 18/07/31 pro4

888888 4201 17/01/01 18/12/31 pro5

999999 6101 16/01/01 17/12/31 pro7

101010 7101 17/08/01 18/07/31 pro9

946537 5101 18/05/01 19/05/01 pro5

\*/

-------------------------------------------------------------------------------------

6)

--Pay amount calculator

--Resident has their own rent fee with adjusted promotion deductive and usage of electricity bill.

--so pay amount could be charge more or less than they expected.

--this function calculate their accurate pay by typing their resident ID.

----------------------------------

create or replace function pay\_cal

(i\_resid number)

return DECIMAL

is

ret\_rate DECIMAL(6,2);

ret\_pro\_deduc NUMBER(5,2);

ret\_ele DECIMAL(6,2);

ret\_pay DECIMAL(6,2);

ret\_res\_coun number;

BEGIN

SELECT RATE, ELE\_BILL, RES\_COUNT

INTO ret\_rate, ret\_ele, ret\_res\_coun

FROM Building B, Contract C, Apartment A

WHERE C.RESID = i\_resid

AND A.APT# = C.APT#

AND A.BUILDING# = B.BUILDING#;

SELECT DEDUCTION INTO ret\_pro\_deduc

FROM Promotion P, Contract C1

WHERE C1.RESID = i\_resid

AND C1.Promo# = P.PROMO#;

--monthly rate has been deducted by promotion deductive

--and then additionally electricity bill devided by number of roommates

ret\_pay :=

ret\_rate \* (1 + ret\_pro\_deduc)

+ (ret\_ele / ret\_res\_coun);

RETURN ret\_pay;

END pay\_cal ;

/

--------------------------------

Set serveroutput on

DECLARE

r\_resid number(10) := '';

a\_pay DECIMAL(6,2):= '';

r\_fname CHAR(12):= '';

r\_lname CHAR(12):= '';

ori\_rate DECIMAL(6,2):= '';

ex\_resid EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_resid,-06502);

BEGIN

r\_resid := '&Enter\_Resident\_ID';

a\_pay := pay\_cal(r\_resid);

SELECT RES\_FNAME, RES\_LNAME

INTO r\_fname, r\_lname

FROM Resident

WHERE RESID = r\_resid;

SELECT RATE

INTO ori\_rate

FROM Building B, Contract C, Apartment A

WHERE C.RESID = r\_resid

AND A.APT# = C.APT#

AND A.BUILDING# = B.BUILDING#;

--display the result

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Thanks for using payment calculator');

DBMS\_OUTPUT.PUT\_LINE('Resident ID : '||r\_resid);

DBMS\_OUTPUT.PUT\_LINE('Name : '||trim(r\_fname)||' '||trim(r\_lname));

DBMS\_OUTPUT.PUT\_LINE('adjusted bill : $'||a\_pay);

DBMS\_OUTPUT.PUT\_LINE('original bill : $'||ori\_rate);

DBMS\_OUTPUT.PUT\_LINE('');

EXCEPTION

WHEN ex\_resid THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Resident ID input error!');

DBMS\_OUTPUT.PUT\_LINE('Try with proper ID');

END;

------------------------

--output check

--when '666666' entered

/\*

Thanks for using payment calculator

Resident ID : 666666

Name : Lewes Kedi

adjusted bill : $578

original bill : $500

\*/

------------------------

--When 'asdf' entered

/\*

Resident ID input error!

Try with proper ID

\*/

7)

--Damaged items total amount calculator

--This helps to see items sperately whether it fixed or not.

--It list which items already fixed and which items need fixed and thier cost.

--each list finished, total amount of each list counted.

--paid amount for fixing and total amount cost would charged for fixing.

----------------------------------------------------------------------

DECLARE

sum\_total DECIMAL(6,2):= 0.00;

cursor cur\_dam IS

SELECT \* FROM DAMAGED ;

BEGIN

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Fixed Items');

DBMS\_OUTPUT.PUT\_LINE ('');

FOR rec\_dam IN cur\_dam

LOOP

IF rec\_dam.FIXED = 'o' then

DBMS\_OUTPUT.PUT\_LINE ('Fixed Item ID : '||trim (rec\_dam.DamID));

DBMS\_OUTPUT.PUT\_LINE ('Apart Number : '||trim (rec\_dam.apt#));

DBMS\_OUTPUT.PUT\_LINE ('Fixed Item : '||trim (rec\_dam.Dam\_Item));

DBMS\_OUTPUT.PUT\_LINE ('Paid Cost : $'||trim (rec\_dam.Dam\_Cost));

DBMS\_OUTPUT.PUT\_LINE ('');

END IF;

END LOOP;

SELECT SUM(DAM\_cost) INTO sum\_total

from DAMAGED

WHERE FIXED = 'o';

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Total amount paid for fixing : $'||sum\_total);

----------------------------------------------------------------

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Damaged Items');

DBMS\_OUTPUT.PUT\_LINE ('');

FOR rec\_dam IN cur\_dam

LOOP

IF rec\_dam.FIXED = 'x' then

DBMS\_OUTPUT.PUT\_LINE ('Damaged Item ID : '||trim (rec\_dam.DamID));

DBMS\_OUTPUT.PUT\_LINE ('Apart Number : '||trim (rec\_dam.apt#));

DBMS\_OUTPUT.PUT\_LINE ('Damaged Item : '||trim (rec\_dam.Dam\_Item));

DBMS\_OUTPUT.PUT\_LINE ('Cost : $'||trim (rec\_dam.Dam\_Cost));

DBMS\_OUTPUT.PUT\_LINE ('');

END IF;

END LOOP;

SELECT SUM(DAM\_cost) INTO sum\_total

from DAMAGED

WHERE FIXED = 'x';

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Total amount cost would charged : $'||sum\_total);

END;

/

----------------------------------------------------------

--output check

/\*

Fixed Items

Fixed Item ID : D1

Apart Number : 1102

Fixed Item : Window

Paid Cost : $100

Fixed Item ID : D2

Apart Number : 2101

Fixed Item : Refregerator

Paid Cost : $250

Fixed Item ID : D3

Apart Number : 2102

Fixed Item : Door

Paid Cost : $80

Fixed Item ID : D4

Apart Number : 5101

Fixed Item : Sink

Paid Cost : $30

Fixed Item ID : D8

Apart Number : 7201

Fixed Item : Couch

Paid Cost : $35

Total amount paid for fixing : $495

----------------------------------------------

Damaged Items

Damaged Item ID : D5

Apart Number : 4101

Damaged Item : Pipe

Cost : $50

Damaged Item ID : D6

Apart Number : 4201

Damaged Item : Pipe

Cost : $50

Damaged Item ID : D7

Apart Number : 6101

Damaged Item : Wall

Cost : $40

Damaged Item ID : D9

Apart Number : 3102

Damaged Item : Window

Cost : $100

Damaged Item ID : D10

Apart Number : 1101

Damaged Item : Dishwasher

Cost : $125

Total amount cost would charged : $365

\*/

----------------------------------------------------------------------------

--8)

--update log for the resident and manager table using trigger

--when some information has updated in each table, a trigger run and store the changed information into the

--update log table. It records when and which table's which row have been updated, and the

--log table also indicates the data of changed attributes that from what value to what value has updated.

--finally a trigger for counting how many rows changed on each table.

-------------------------------------------------------------------

--DROP/CREATE update log table

DROP TABLE res\_up\_log;

CREATE TABLE res\_up\_log

(date\_time timestamp(6),

Updated\_table varchar2(20),

Updated\_row varchar2(20),

From\_what varchar2(30),

To\_what varchar2(30));

DROP TABLE res\_up\_log\_count;

CREATE TABLE res\_up\_log\_count

(updated\_table varchar2(20),

Number\_of\_update number(3));

---------------------

--create dummy table

DROP TABLE Resident5;

CREATE TABLE Resident5

as SELECT \* FROM Resident;

----------------------

DROP TABLE Manager5;

CREATE TABLE Manager5

as SELECT \* FROM Manager;

/

----------------------------------

--create Resident Fname update trigger

create or replace TRIGGER tri\_up\_res\_Fname

AFTER UPDATE OF Res\_Fname ON Resident5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table,

Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Resident', 'Res\_Fname',

:Old.Res\_Fname, :New.Res\_Fname);

END;

/

--create Resident Lname update trigger

create or replace TRIGGER tri\_up\_res\_Lname

AFTER UPDATE OF Res\_Lname ON Resident5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table,

Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Resident', 'Res\_Lname',

:Old.Res\_Lname, :New.Res\_Lname);

END;

/

--create Resident phone update trigger

create or replace TRIGGER tri\_up\_res\_phone

AFTER UPDATE OF Res\_phone ON Resident5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table

,Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Resident', 'Res\_phone',

:Old.Res\_phone, :New.Res\_phone);

END;

/

--create Resident email update trigger

create or replace TRIGGER tri\_up\_res\_email

AFTER UPDATE OF Res\_email ON Resident5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table,

Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Resident', 'Res\_email',

:Old.Res\_email, :New.Res\_email);

END;

/

-----------------------------------

--create Manager Fname update trigger

create or replace TRIGGER tri\_up\_mg\_Fname

AFTER UPDATE OF Mgr\_Fname ON Manager5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table,

Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Manager', 'Mgr\_Fname',

:Old.Mgr\_Fname, :New.Mgr\_Fname);

END;

/

--create Manager Lname update trigger

create or replace TRIGGER tri\_up\_mg\_Lname

AFTER UPDATE OF Mgr\_Lname ON Manager5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table,

Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Manager', 'Mgr\_Lname',

:Old.Mgr\_Lname, :New.Mgr\_Lname);

END;

/

--create Manager phone update trigger

create or replace TRIGGER tri\_up\_mg\_phone

AFTER UPDATE OF Mgr\_phone ON Manager5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table,

Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Manager', 'Mgr\_phone',

:Old.Mgr\_phone, :New.Mgr\_phone);

END;

/

--create Manager email update trigger

create or replace TRIGGER tri\_up\_mg\_email

AFTER UPDATE OF Mgr\_email ON Manager5

FOR EACH ROW

BEGIN

INSERT INTO

res\_up\_log(date\_time,Updated\_table,

Updated\_row,From\_what,To\_what)

VALUES

(sysdate, 'Manager', 'Mgr\_email',

:Old.Mgr\_email, :New.Mgr\_email);

END;

/

---------------------------------------

create or replace TRIGGER tri\_up\_count

AFTER INSERT ON res\_up\_log

DECLARE

Num\_update\_res number := 0;

Num\_update\_mgr number := 0;

BEGIN

DELETE FROM res\_up\_log\_count;

SELECT COUNT(\*) INTO Num\_update\_res

FROM res\_up\_log WHERE Updated\_table = 'Resident';

SELECT COUNT(\*) INTO Num\_update\_mgr

FROM res\_up\_log WHERE Updated\_table = 'Manager';

INSERT INTO res\_up\_log\_count

(updated\_table, Number\_of\_update)

VALUES

('Resident',Num\_update\_res);

INSERT INTO res\_up\_log\_count

(updated\_table, Number\_of\_update)

VALUES

('Manager',Num\_update\_mgr);

END;

/

----------------------------------------

UPDATE Resident5 SET Res\_Fname = 'Kammel'

WHERE ResID = '222222';

UPDATE Resident5 SET Res\_Lname = 'Crain'

WHERE ResID = '333333';

UPDATE Resident5 SET Res\_phone = '662-124-8745'

WHERE ResID = '444444';

UPDATE Resident5 SET Res\_email = 'owie@gmail.com'

WHERE ResID = '555555';

UPDATE Resident5 SET Res\_Fname = 'Oliber'

WHERE ResID = '666666';

-----------------------

UPDATE Manager5 SET Mgr\_Fname = 'Braian'

WHERE MgrID = 'M321321';

UPDATE Manager5 SET Mgr\_Lname = 'Gill'

WHERE MgrID = 'M456456';

UPDATE Manager5 SET Mgr\_phone = '662-553-7733'

WHERE MgrID = 'M654654';

UPDATE Manager5 SET Mgr\_email = 'MGwiil@gmail.com'

WHERE MgrID = 'M789789';

SELECT \* FROM Resident5;

SELECT \* FROM Manager5;

SELECT \* FROM res\_up\_log;

SELECT \* FROM res\_up\_log\_count;

------------------------------------------------------------------------------

--output

/\*

-- update\_log table

DATE\_TIME UPDATED\_TABLE UPDATED\_ROW FROM\_WHAT TO\_WHAT

--------------------------- -------------------- -------------------- ------------------------------ ------------------------------

18/05/06 03:30:01.000000000 Resident Res\_Fname Tim Kammel

18/05/06 03:30:01.000000000 Resident Res\_Lname Tong Crain

18/05/06 03:30:01.000000000 Resident Res\_phone 662-444-4444 662-124-8745

18/05/06 03:30:01.000000000 Resident Res\_email ttt@google.com owie@gmail.com

18/05/06 03:30:01.000000000 Resident Res\_Fname Lewes Oliber

18/05/06 03:30:01.000000000 Manager Mgr\_Fname Susan Braian

18/05/06 03:30:01.000000000 Manager Mgr\_Lname Kale Gill

18/05/06 03:30:01.000000000 Manager Mgr\_phone 662-654-6543 662-553-7733

18/05/06 03:30:01.000000000 Manager Mgr\_email xxx@google.com MGwiil@gmail.com

------------------------------------------------------------------------------------

--log counter table

UPDATED\_TABLE NUMBER\_OF\_UPDATE

-------------------- ----------------

Resident 5

Manager 4

\*/

----------------------------------------------------------------------------------------------

--9)

--user input specific month( ex) ’10’ or ‘Oct’) then resident’s information and paid date shows who have

--paid for that month.

SET serveroutput ON;

DECLARE

mon varchar2(15) := '&Enter\_month\_\_\_EX\_`Oct`\_OR\_`10`';

CURSOR CUR IS

SELECT RES.ResID, RES.Res\_Fname, REN.Paid\_date

FROM Rent\_pay REN, Resident RES

WHERE REN.ResID = RES.ResID;

--------------------------------

BEGIN

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Resident who paid the rent of '||mon);

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Resident ID First Name Paid Date');

DBMS\_OUTPUT.PUT\_LINE ('------------- ------------ -----------');

-----------------------

IF mon = '01' OR mon = 'Jen' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-01-31'

AND REC.Paid\_date >= '2017-01-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

-----------------------

ELSIF mon = '02' OR mon = 'Feb' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-02-28'

AND REC.Paid\_date >= '2017-02-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

------------------------

ELSIF mon = '03' OR mon = 'Mar' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-03-31'

AND REC.Paid\_date >= '2017-03-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

------------------------

ELSIF mon = '04' OR mon = 'Apr' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-04-30'

AND REC.Paid\_date >= '2017-04-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

------------------------

ELSIF mon = '05' OR mon = 'May' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-05-31'

AND REC.Paid\_date >= '2017-05-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

------------------------

ELSIF mon = '06' OR mon = 'Jun' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-06-30'

AND REC.Paid\_date >= '2017-06-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

---------------------

ELSIF mon = '07' OR mon = 'Jul' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-07-31'

AND REC.Paid\_date >= '2017-07-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

----------------------

ELSIF mon = '08' OR mon = 'Aug' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-08-31'

AND REC.Paid\_date >= '2017-08-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

---------------------

ELSIF mon = '09' OR mon = 'Sep' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-09-30'

AND REC.Paid\_date >= '2017-09-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

---------------------

ELSIF mon = '10' OR mon = 'Sep' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-10-31'

AND REC.Paid\_date >= '2017-10-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

----------------------

ELSIF mon = '11' OR mon = 'Nov' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-11-30'

AND REC.Paid\_date >= '2017-11-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

----------------------

ELSIF mon = '12' OR mon = 'Dec' THEN

FOR REC IN CUR

LOOP

IF REC.Paid\_date <= '2017-12-31'

AND REC.Paid\_date >= '2017-12-01' THEN

DBMS\_OUTPUT.PUT\_LINE (' '||REC.RESID||' '||REC.RES\_FNAME||' '||REC.PAID\_DATE);

END IF;

END LOOP;

END IF;

-------------------------

END;

/

---------------------------------------------------------

--output

/\*

--input 'Oct'

Resident who paid the rent of Oct

Resident ID First Name Paid Date

222222 Tim 17/10/01

444444 Sandie 17/10/02

777777 George 17/10/02

--------------------------

--input '11'

Resident who paid the rent of 11

Resident ID First Name Paid Date

111111 Krag 17/11/04

222222 Tim 17/11/01

333333 Kenny 17/11/02

555555 Kim 17/11/02

666666 Lewes 17/11/02

777777 George 17/11/03

888888 Marie 17/11/03

------------------------------------------------------------------------------------------------------

--10)

--Facility total cost calculator grouped by each department, and the date.

-------------------------------------------

SET serveroutput ON;

DECLARE

CURSOR CUR IS

SELECT Dep, SUM(Fac\_cost) Total\_cost

FROM Facility

GROUP BY Dep;

CURSOR CUR1 IS

SELECT Acquision\_date, SUM(Fac\_cost) Total\_cost

FROM Facility

GROUP BY Acquision\_date;

BEGIN

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Total Facility Cost of each Department');

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE (' Department Total Cost');

DBMS\_OUTPUT.PUT\_LINE ('------------ -------------');

FOR REC IN CUR

LOOP

DBMS\_OUTPUT.PUT\_LINE (' '||LPAD(REC.DEP,10)

||' $'||trim(REC.Total\_cost));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('-----------------------------------------');

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE ('Total Facility Cost on the date ');

DBMS\_OUTPUT.PUT\_LINE ('');

DBMS\_OUTPUT.PUT\_LINE (' Acquision\_date Total Cost');

DBMS\_OUTPUT.PUT\_LINE ('--------------- -----------');

FOR REC IN CUR1

LOOP

DBMS\_OUTPUT.PUT\_LINE (' '||LPAD(REC.Acquision\_date,9)

||' $'||trim(REC.Total\_cost));

END LOOP;

END;

/

--output

/\*

Total Facility Cost of each Department

Department Total Cost

------------ -------------

Janitor $150

Office $1450

Frontdesk $1075

Guestroom $1350

-----------------------------------------

Total Facility Cost on the date

Acquision\_date Total Cost

--------------- -----------

16/08/10 $200

15/07/25 $2000

16/08/16 $150

16/07/05 $650

16/03/13 $1025

\*/

------------------------------------------------------------------------------------------------

--11)

--calculate the average electric bill first and find the unit and resident who

-- charged more than average electric bill

-------------------------------------------

SET serveroutput ON;

DECLARE

AVG\_B INTEGER(5,2);

CURSOR CUR IS

Select Res\_Fname, Res\_Lname, A1.apt#, A1.Ele\_Bill

From Resident R1, Contract C1, apartment A1,

(select avg(Ele\_Bill) ELE\_AVG

from Apartment) AVGA

Where A1.apt# in

(Select apt#

from apartment

where Ele\_Bill != 0)

AND A1.Ele\_Bill >= AVGA.ELE\_AVG

AND R1.ResID = C1.ResID

AND A1.APT# = C1.APT#;

----------------------------

BEGIN

select avg(Ele\_Bill)

INTO AVG\_B

from Apartment;

DBMS\_OUTPUT.PUT\_LINE('Average Electric bill : $'||AVG\_B);

DBMS\_OUTPUT.PUT\_LINE('Apartments that more charged than average for Electric bill:');

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE(' Resident name APT# Elec\_Bill');

DBMS\_OUTPUT.PUT\_LINE(' ------------------ ---- ---------');

FOR REC IN CUR

LOOP

DBMS\_OUTPUT.PUT\_LINE(' '||lpad(REC.Res\_Fname,10)||

''||lpad(REC.Res\_Lname,10)||

''||lpad(REC.apt#,6)||' $'||rpad(REC.Ele\_Bill,4));

END LOOP;

END;

/

-----------------------------------------------------------------

--output

/\*

Average Electric bill : $46.43

Apartments that more charged than average for Electric bill:

Resident name APT# Elec\_Bill

------------------ ---- ---------

Kenny Tong 2101 $80

Sandie Gardner 2101 $80

Kim Baker 2101 $80

Lewes Kedi 3102 $55

George Hal 3102 $55

\*/

-------------------------------------------------------------------------------------

--12)

--Login process

--Manager enters their email address and password to access

--once they loged in, their departmant and hotline number showen.

SET serveroutput ON;

DECLARE

I\_email VARCHAR2(20);

I\_pass VARCHAR2(25);

Mfname VARCHAR2(20);

MID VARCHAR2(10);

DEPA VARCHAR2(15);

HOTL NUMBER(5);

ex\_val EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_val,-06502);

BEGIN

I\_email := '&Enter\_login\_email'; -- ddd@google.com

I\_pass := '&Enter\_Password'; --ddd

Select Mgr\_Fname, MGRID

INTO Mfname, MID

FROM Manager

WHERE Mgr\_email = I\_email

AND Mgr\_password = I\_pass;

DBMS\_OUTPUT.PUT\_LINE('');

IF Mfname is not null

THEN

DBMS\_OUTPUT.PUT\_LINE('Valid User');

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('Welcome '||trim(Mfname)||'!');

SELECT Dep, Hotline\_num

INTO DEPA, HOTL

FROM Departments

WHERE MgrID = MID;

DBMS\_OUTPUT.PUT\_LINE('Department: '||DEPA);

DBMS\_OUTPUT.PUT\_LINE('HotLine number: '||HOTL);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Invalid User!');

DBMS\_OUTPUT.PUT\_LINE('Try again');

END IF;

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('technical support center : 662-742-7515');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('No data found Or Invalid User!');

DBMS\_OUTPUT.PUT\_LINE('Try again!');

WHEN ex\_val THEN

DBMS\_OUTPUT.PUT\_LINE('');

DBMS\_OUTPUT.PUT\_LINE('improper input!');

DBMS\_OUTPUT.PUT\_LINE('Try again!');

END;

/

------------------------------------------------------------------

--output

/\*

--input ‘ddd@google.com’ , ‘ddd’

Valid User

Welcome Jim!

Department: Building15

HotLine number: 103

technical support center : 662-742-7515

---------------------------------------------------

--input ‘zzz@google.com’ , ‘zzz’

Valid User

Welcome Anna!

Department: Office

HotLine number: 102

technical support center : 662-742-7515

--------------------------------------------------------

--input ‘A@(#RUFI’ , ‘::KFA@#IH’ – just random values

No data found Or Invalid User!

Try again!

\*/