

## Project for Database Design

# *Phase III. Implementation*

Indhumadhi Suryanarayanan  
[ixs150630@utdallas.edu](mailto:ixs150630@utdallas.edu)

Sindhujaa Ranganathan Chandra Babu  
[sxr152530@utdallas.edu](mailto:sxr152530@utdallas.edu)

Suneesha Kudipudi  
[sxk157430@utdallas.edu](mailto:sxk157430@utdallas.edu)

(Week 11-15: Nov.1-Nov.30)

## 0. Pre-Illumination

For clearly describing the implementation of our database, we separate this report into four sections. In Section 1 we normalize the original relational schema into third normal form and changed part of our relational schema because of some requirement from Phase III. We then explained what are changed. In Section 2 we drew a dependency diagram for each relation table one by one. In Section 3 we began our process of building a database in Oracle using SQL statements, which contains three parts. Part one is the creation of database, including tables, all other structures as well as data type and format, Part two is the creation of views corresponding to five distinct requirements from Question d, and Part three is the creation of Queries to satisfy 14 requirements from Question e. Finally, a short summary is given at the end of this report.

## 1. Modified Relational Schema

The table SUBDIVISION has to be normalized to 3NF since it had a transitive dependency of NSME->MANAGER\_ID, MANAGER\_ID->START\_DATE.

Hence it had to be decomposed as follows:

### SUBDIVISION:

<u>NAME</u>	START_YEAR	RANK	ZIP	MANAGER_ID
-------------	------------	------	-----	------------

### MANAGER:

<u>MANAGER_ID</u>	START_DATE
-------------------	------------

Figure 1: 3NF Normalized tables

Rests of the tables remain the same.

## 2. Dependency Diagram

We now draw a dependency diagram for each tables in the below sections.

### 2.1 Dependency in MANAGER

MANAGER:

<u>MANAGER_ID</u>	START_DATE
-------------------	------------

### 2.2 Dependency in LOT

LOT:

<u>SUBDIVISION_NAME</u>	<u>NUMBER</u>	SIZE	FACING	AVAILABILITY
-------------------------	---------------	------	--------	--------------

### 2.3 Dependency in FLOORPLAN

FLOORPLAN:

<u>SUBDIVISION_NAME</u>	<u>NUMBER</u>	FLOOR_COUNT	BED_COUNT	BATH_COUNT	NAME	PRICE	SQ_FT	GARAGE_COUNT
-------------------------	---------------	-------------	-----------	------------	------	-------	-------	--------------

### 2.4 Dependency in INVENTORY

INVENTORY:

<u>SUBDIVISION_NAME</u>	<u>ID</u>	PRICE	MOVE_IN_DATE
-------------------------	-----------	-------	--------------

### 2.5 Dependency in SUBDIVISION

**SUBDIVISION:**

<u>NAME</u>	START_YEAR	RANK	ZIP	MANAGER_ID
-------------	------------	------	-----	------------

**2.6 Dependency in TOURS****TOURS:**

<u>EMP_ID</u>	<u>CUSTOMER_EMAIL</u>	<u>I_SUBDIVISION_NAME</u>	<u>INVENTORY_ID</u>	TOUR_DATE
---------------	-----------------------	---------------------------	---------------------	-----------

**2.7 Dependency in ASSOCIATES****ASSOCIATES:**

<u>L_SUBDIVISION_NAME</u>	<u>LOT_NUMBER</u>	<u>F_SUBDIVISION_NAME</u>	<u>FLOORPLAN_NUMBER</u>	CONTRACT_ID
---------------------------	-------------------	---------------------------	-------------------------	-------------

**2.8 Dependency in EMPLOYEE****EMPLOYEE:**

<u>ID</u>	FNAME	LNAME	YEARS_OF_EXP	E_TYPE	SALARY	SUBDIVISION_NAME	LOANAPPLICANT_ID
-----------	-------	-------	--------------	--------	--------	------------------	------------------

**2.9 Dependency in SECRETARY****SECRETARY:**

<u>EMP_ID</u>	HOURLY_WAGE
---------------	-------------

**2.10 Dependency in VISITS****VISITS:**

<u>CUSTOMER_EMAIL</u>	<u>SUBDIVISION_NAME</u>	VISIT_DATE
-----------------------	-------------------------	------------

**2.11 Dependency in SIGNS****SIGNS:**

<u>EXISTING_CUSTOMER_EMAIL</u>	<u>CONTRACT_ID</u>	<u>SUBDIVISION_NAME</u>
--------------------------------	--------------------	-------------------------

## 2.12 Dependency in CONTRACT

### CONTRACT:

<u>ID</u>	SIGN_DATE	SALE_PRICE	EMPLOYEE_ID
-----------	-----------	------------	-------------

## 2.13 Dependency in CUSTOMER

### CUSTOMER:

<u>EMAIL</u>	NAME	INCOME	CREDIT_SCORE	BATH_COUNT	TYPE	BED_COUNT	SQ_FT	RANK	LOANAPPLICANT_ID
--------------	------	--------	--------------	------------	------	-----------	-------	------	------------------

## 2.14 Dependency in LOAN

### LOAN:

<u>ID</u>	TERM	TYPE	DATE	STATUS	CREDIT_REPORT	LOANAPPLICANT_ID
-----------	------	------	------	--------	---------------	------------------

## 2.15 Final Results

Figure 2.15 shows the final results for the whole database including the ones who do not have any functional dependencies.

### MANAGER:

<u>MANAGER ID</u>	START_DATE
-------------------	------------

### LOT:

<u>SUBDIVISION NAME</u>	<u>NUMBER</u>	SIZE	FACING	AVAILABILITY
-------------------------	---------------	------	--------	--------------

### FLOORPLAN:

<u>SUBDIVISION NAME</u>	<u>NUMBER</u>	FLOOR_COUNT	BED_COUNT	BATH_COUNT	NAME	PRICE	SQ_FT	GARAGE_COUNT
-------------------------	---------------	-------------	-----------	------------	------	-------	-------	--------------

**INVENTORY:**

<u>SUBDIVISION NAME</u>	<u>ID</u>	PRICE	MOVE_IN_DATE
-------------------------	-----------	-------	--------------

**SUBDIVISION:**

<u>NAME</u>	START_YEAR	RANK	ZIP	MANAGER_ID
-------------	------------	------	-----	------------

**TOURS:**

<u>EMP ID</u>	<u>CUSTOMER EMAIL</u>	<u>I SUBDIVISION NAME</u>	<u>INVENTORY ID</u>	TOUR_DATE
---------------	-----------------------	---------------------------	---------------------	-----------

**ASSOCIATES:**

<u>L SUBDIVISION NAME</u>	<u>LOT NUMBER</u>	<u>F SUBDIVISION NAME</u>	<u>FLOORPLAN NUMBER</u>	CONTRACT_ID
---------------------------	-------------------	---------------------------	-------------------------	-------------

**EMPLOYEE:**

<u>ID</u>	FNAME	LNAME	YEARS_OF_EXP	E_TYPE	SALARY	SUBDIVISION_NAME	LOANAPPLICANT_ID
-----------	-------	-------	--------------	--------	--------	------------------	------------------

**SECRETARY:**

<u>EMP ID</u>	HOURLY_WAGE
---------------	-------------

**VISITS:**

<u>CUSTOMER EMAIL</u>	<u>SUBDIVISION NAME</u>	VISIT_DATE
-----------------------	-------------------------	------------

**SIGNS:**

<u>EXISTING CUSTOMER EMAIL</u>	<u>CONTRACT ID</u>	<u>SUBDIVISION NAME</u>
--------------------------------	--------------------	-------------------------

**CONTRACT:**

<u>ID</u>	SIGN_DATE	SALE_PRICE	EMPLOYEE_ID
-----------	-----------	------------	-------------

**CUSTOMER:**

<u>EMAIL</u>	NAME	INCOME	CREDIT_SCORE	BATH_COUNT	TYPE	BED_COUNT	SQ_FT	RANK	LOANAPPLICANT_ID
--------------	------	--------	--------------	------------	------	-----------	-------	------	------------------

**LOAN:**

<u>ID</u>	TERM	TYPE	DATE	STATUS	CREDIT_REPORT	LOANAPPLICANT_ID
-----------	------	------	------	--------	---------------	------------------

**EXISTING:**

<u>EMAIL</u>
--------------

**POTENTIAL:**

<u>EMAIL</u>
--------------

**LOANAPPLICANT:**

<u>ID</u>
-----------

**CUSTOMERPHONE:**

<u>EMAIL</u>	<u>PHONE</u>
--------------	--------------

**SALESAGENT:**

<u>EMPLOYEE ID</u>	<u>SHIFT</u>
--------------------	--------------

**EMPLOYEEPHONE:**

<u>EMPLOYEE ID</u>	<u>PHONE</u>
--------------------	--------------

**COMES:**

<u>I SUBDIVISION NAME</u>	<u>INVENTORY ID</u>	<u>L SUBDIVISION NAME</u>	<u>LOT NUMBER</u>	<u>F SUBDIVISION NAME</u>	<u>FLOORPLAN NUMBER</u>
---------------------------	---------------------	---------------------------	-------------------	---------------------------	-------------------------

Figure 2.15 : Whole Dependency Diagram for Home Builder Company Database

## 3. Implementation of Database

### 3.1 Creation of Database with SQL Statements

After normalizing every relational schema into third normal form and modifying some

details, it is the time to implement our database using SQL languages into Oracle.

### 3.1.1 Table Creation

Using SQL statement, we created tables as follows:

```
CREATE TABLE MANAGER (MANAGER_ID INT NOT NULL, START_DATE DATE,  
CONSTRAINT MANAGER_PK PRIMARY KEY (MANAGER_ID));
```

```
CREATE TABLE SUBDIVISION(NAME VARCHAR(255) NOT NULL, START_YEAR  
INT ,RANK INT, ZIP INT, MANAGER_ID INT);
```

```
ALTER TABLE SUBDIVISION ADD CONSTRAINT PK_SUB PRIMARY KEY (NAME);
```

```
ALTER TABLE SUBDIVISION ADD CONSTRAINT FK_MANAGER FOREIGN KEY  
(MANAGER_ID) REFERENCES MANAGER (MANAGER_ID) ON DELETE CASCADE  
INITIALLY DEFERRED;
```

```
ALTER TABLE SUBDIVISION ADD CONSTRAINT CK_MAN UNIQUE(MANAGER_ID);
```

```
CREATE TABLE LOANAPPLICANT (ID INT NOT NULL, CONSTRAINT LOANAPP_PK  
PRIMARY KEY (ID));
```

```
CREATE TABLE LOT (SUBDIVISION_NAME VARCHAR(255), LOT_NUMBER INT UNIQUE,  
LOT_SIZE INT, FACING VARCHAR(20) CHECK (FACING IN('NORTH','SOUTH','EAST',  
'WEST')), AVAILABILITY VARCHAR(5) CHECK (AVAILABILITY IN ('YES', 'NO')));
```

```
ALTER TABLE LOT ADD CONSTRAINT SUB_FK FOREIGN KEY (SUBDIVISION_NAME)  
REFERENCES SUBDIVISION(NAME) ON DELETE CASCADE INITIALLY DEFERRED;
```

```
ALTER TABLE LOT ADD CONSTRAINT LOT_PK PRIMARY KEY (SUBDIVISION_NAME,  
LOT_NUMBER);
```

```
CREATE TABLE FLOORPLAN (SUBDIVISION_NAME VARCHAR(255), FP_NUMBER  
NUMBER(4,0) ,  
FLOOR_COUNT INT CHECK(FLOOR_COUNT>=1 AND FLOOR_COUNT<=3), BED_COUNT  
INT CHECK(BED_COUNT>=1 AND BED_COUNT<=5), BATH_COUNT INT  
CHECK(BATH_COUNT>=1 AND BATH_COUNT<=5), NAME VARCHAR(255), PRICE INT,  
SQ_FT NUMBER(4,0), GARAGE_COUNT INT CHECK(GARAGE_COUNT>=1 AND  
GARAGE_COUNT<=3));
```

```
ALTER TABLE FLOORPLAN ADD CONSTRAINT FP_FK FOREIGN KEY  
(SUBDIVISION_NAME) REFERENCES SUBDIVISION(NAME) ON DELETE CASCADE  
INITIALLY DEFERRED;
```

```
ALTER TABLE FLOORPLAN ADD CONSTRAINT FP_PK PRIMARY KEY  
(SUBDIVISION_NAME, FP_NUMBER);
```

```
CREATE TABLE INVENTORY(SUBDIVISION_NAME VARCHAR(255), ID INT, PRICE INT,  
MOVE_IN_DATE DATE);
```

```
ALTER TABLE INVENTORY ADD CONSTRAINT INVENT_FK FOREIGN KEY  
(SUBDIVISION_NAME) REFERENCES SUBDIVISION(NAME) ON DELETE CASCADE  
INITIALLY DEFERRED;
```

```
ALTER TABLE INVENTORY ADD CONSTRAINT IN_PK PRIMARY KEY
(SUBDIVISION_NAME, ID);
```

```
CREATE TABLE COME( I_SUBDIVISION_NAME VARCHAR(255), INVENTORY_ID INT,
L_SUBDIVISION_NAME VARCHAR(255), LOT_NUMBER INT, F_SUBDIVISION_NAME
VARCHAR(255), FLOORPLAN_NUMBER INT);
```

```
ALTER TABLE COME ADD CONSTRAINT IN_COME_FK FOREIGN KEY
(I_SUBDIVISION_NAME,INVENTORY_ID) REFERENCES
INVENTORY(SUBDIVISION_NAME,ID) ON DELETE CASCADE INITIALLY DEFERRED;
```

```
ALTER TABLE COME ADD CONSTRAINT L_COME_FK FOREIGN KEY
(L_SUBDIVISION_NAME,LOT_NUMBER) REFERENCES
LOT(SUBDIVISION_NAME,LOT_NUMBER) ON DELETE CASCADE INITIALLY DEFERRED;
```

```
ALTER TABLE COME ADD CONSTRAINT F_COME_FK FOREIGN KEY
(F_SUBDIVISION_NAME,FLOORPLAN_NUMBER) REFERENCES
FLOORPLAN(SUBDIVISION_NAME,FP_NUMBER) ON DELETE CASCADE INITIALLY
DEFERRED;
```

```
ALTER TABLE COME ADD CONSTRAINT COME_PK PRIMARY KEY
( I_SUBDIVISION_NAME, INVENTORY_ID, L_SUBDIVISION_NAME, LOT_NUMBER,
F_SUBDIVISION_NAME, FLOORPLAN_NUMBER);
```

```
CREATE TABLE EMPLOYEE (ID INT, FNAME VARCHAR(255), LNAME VARCHAR(255),
YEARS_OF_EXPERIENCE INT CHECK(YEARS_OF_EXPERIENCE>=0 AND
YEARS_OF_EXPERIENCE<=55), E_TYPE VARCHAR(20) CHECK (E_TYPE IN
('SECRETARY','SALES_AGENT')), SALARY DECIMAL(10,2), SUBDIVISION_NAME
VARCHAR(255), LOANAPPLICANT_ID INT);
```

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT EM_SUB_FK FOREIGN KEY
(SUBDIVISION_NAME) REFERENCES SUBDIVISION(NAME) ON DELETE CASCADE
INITIALLY DEFERRED;
```

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT EM_LA_FK FOREIGN KEY
(LOANAPPLICANT_ID) REFERENCES LOANAPPLICANT(ID) ON DELETE CASCADE
INITIALLY DEFERRED;
```

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT EMPLOY_PK PRIMARY KEY (ID);
```

```
CREATE TABLE EMPLOYEE (ID INT, FNAME VARCHAR(255), LNAME VARCHAR(255),
YEARS_OF_EXPERIENCE INT CHECK(YEARS_OF_EXPERIENCE>=0 AND
YEARS_OF_EXPERIENCE<=55), E_TYPE VARCHAR(20) CHECK (E_TYPE IN
('SECRETARY','SALES_AGENT')), SALARY DECIMAL(10,2), SUBDIVISION_NAME
VARCHAR(255), LOANAPPLICANT_ID INT);
```

```
CREATE TABLE CONTRACT (ID INT , SIGN_DATE DATE, SALE_PRICE INT ,EMPLOYEE_ID
INT);
```

```
ALTER TABLE CONTRACT ADD CONSTRAINT CON_FK FOREIGN KEY (EMPLOYEE_ID)
REFERENCES EMPLOYEE(ID) ON DELETE CASCADE INITIALLY DEFERRED;
```

```
ALTER TABLE CONTRACT ADD CONSTRAINT CON_PK PRIMARY KEY (ID);
```

```
CREATE TABLE ASSOCIATES(L_SUBDIVISION_NAME VARCHAR(255), LOT_NUMBER INT,
```



```
F_SUBDIVISION_NAME VARCHAR(255), FLOORPLAN_NUMBER INT ,CONTRACT_ID INT );
```

```
ALTER TABLE ASSOCIATES ADD CONSTRAINT L_AS_FK FOREIGN KEY
(L_SUBDIVISION_NAME,LOT_NUMBER) REFERENCES
LOT(SUBDIVISION_NAME,LOT_NUMBER) ON DELETE CASCADE INITIALLY DEFERRED;
```

```
ALTER TABLE ASSOCIATES ADD CONSTRAINT F_AS_FK FOREIGN KEY
(F_SUBDIVISION_NAME,FLOORPLAN_NUMBER) REFERENCES
FLOORPLAN(SUBDIVISION_NAME,FP_NUMBER) ON DELETE CASCADE INITIALLY
DEFERRED;
```

```
ALTER TABLE ASSOCIATES ADD CONSTRAINT CON_AS_FK FOREIGN KEY
(CONTRACT_ID) REFERENCES CONTRACT(ID) ON DELETE CASCADE INITIALLY
DEFERRED;
```

```
ALTER TABLE ASSOCIATES ADD CONSTRAINT AS_PK PRIMARY KEY
( L_SUBDIVISION_NAME, LOT_NUMBER, F_SUBDIVISION_NAME,
FLOORPLAN_NUMBER);
```

```
CREATE TABLE CUSTOMER (EMAIL VARCHAR(255) ,NAME VARCHAR(255) , INCOME
DECIMAL(10,2), CREDIT_SCORE INT CHECK(CREDIT_SCORE>=300 AND
CREDIT_SCORE<=850), BATH_COUNT INT CHECK(BATH_COUNT>=1 AND
BATH_COUNT<=5),TYPE VARCHAR(20) CHECK(TYPE IN
('POTENTIAL','EXISTING')),BED_COUNT INT CHECK(BED_COUNT>=1 AND
BED_COUNT<=5),SQ_FT NUMBER(4,0),RANK INT ,LOANAPPLICANT_ID INT);
```

```
ALTER TABLE CUSTOMER ADD CONSTRAINT CUS_LA_FK FOREIGN KEY
(LOANAPPLICANT_ID) REFERENCES LOANAPPLICANT(ID) ON DELETE CASCADE
INITIALLY DEFERRED;
```

```
ALTER TABLE CUSTOMER ADD CONSTRAINT CUS_PK PRIMARY KEY (EMAIL);
```

```
CREATE TABLE TOURS (EMP_ID INT,CUSTOMER_EMAIL VARCHAR(255),
I_SUBDIVISION_NAME VARCHAR(255), INVENTORY_ID INT, TOUR_DATE DATE) ;
```

```
ALTER TABLE TOURS ADD CONSTRAINT TOURS_EMP_FK FOREIGN KEY (EMP_ID)
REFERENCES EMPLOYEE(ID) ON DELETE CASCADE INITIALLY DEFERRED;
```

```
ALTER TABLE TOURS ADD CONSTRAINT TOURS_CUS_FK FOREIGN KEY
(CUSTOMER_EMAIL) REFERENCES CUSTOMER(EMAIL) ON DELETE CASCADE
INITIALLY DEFERRED;
```

```
ALTER TABLE TOURS ADD CONSTRAINT TOUR_IN_FK FOREIGN KEY
(I_SUBDIVISION_NAME,INVENTORY_ID) REFERENCES
INVENTORY(SUBDIVISION_NAME,ID) ON DELETE CASCADE INITIALLY DEFERRED;
```

```
ALTER TABLE TOURS ADD CONSTRAINT TOURS_PK PRIMARY KEY
(EMP_ID,CUSTOMER_EMAIL,I_SUBDIVISION_NAME,INVENTORY_ID);
```

```
CREATE TABLE EMPLOYEEPHONE (EMPLOYEE_ID INT , PHONE VARCHAR(13));
```

```
ALTER TABLE EMPLOYEEPHONE ADD CONSTRAINT PHONE_EMP_FK FOREIGN KEY
(EMPLOYEE_ID) REFERENCES EMPLOYEE(ID) ON DELETE CASCADE INITIALLY
DEFERRED;
```

```
ALTER TABLE EMPLOYEEPHONE ADD CONSTRAINT EM_PH_PK PRIMARY KEY
```

(EMPLOYEE\_ID,PHONE);

Alter table EMPLOYEEPHONE add constraint "PHONENO\_CHECK" CHECK (REGEXP\_LIKE(PHONE, '([0-9]{3})[0-9]{3}-[0-9]{4}'));

CREATE TABLE SECRETARY (EMPLOYEE\_ID INT, HOURLY\_WAGE DECIMAL(5,2));

ALTER TABLE SECRETARY ADD CONSTRAINT SEC\_EMP\_FK FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEE(ID) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE SECRETARY ADD CONSTRAINT SEC\_PK PRIMARY KEY (EMPLOYEE\_ID);

CREATE TABLE SALES\_AGENT (EMPLOYEE\_ID INT, SALES\_SHIFT VARCHAR(3) );

ALTER TABLE SALES\_AGENT ADD CONSTRAINT NE CHECK(SALES\_SHIFT IN ('monday','tuesday','wednesday','thursday','friday','saturday','sunday'));

ALTER TABLE SALES\_AGENT MODIFY (SALES\_SHIFT VARCHAR(10));

ALTER TABLE SALES\_AGENT ADD CONSTRAINT SALES\_EMP\_FK FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEE(ID) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE SALES\_AGENT ADD CONSTRAINT SALES\_PK PRIMARY KEY (EMPLOYEE\_ID, SALES\_SHIFT);

CREATE TABLE VISITS (CUSTOMER\_EMAIL VARCHAR(255),SUBDIVISION\_NAME VARCHAR(255), VISIT\_DATE DATE);

ALTER TABLE VISITS ADD CONSTRAINT VISITS\_CUS\_FK FOREIGN KEY (CUSTOMER\_EMAIL) REFERENCES CUSTOMER(EMAIL) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE VISITS ADD CONSTRAINT VISIT\_SUB\_FK FOREIGN KEY (SUBDIVISION\_NAME) REFERENCES SUBDIVISION(NAME) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE VISITS ADD CONSTRAINT VIS\_PK PRIMARY KEY (CUSTOMER\_EMAIL,SUBDIVISION\_NAME);

CREATE TABLE EXISTING( EMAIL VARCHAR(255), CONSTRAINT EX\_FK FOREIGN KEY (EMAIL) REFERENCES CUSTOMER(EMAIL) ON DELETE CASCADE INITIALLY DEFERRED, CONSTRAINT EX\_PK PRIMARY KEY(EMAIL));

CREATE TABLE POTENTIAL( EMAIL VARCHAR(255), CONSTRAINT PO\_FK FOREIGN KEY (EMAIL) REFERENCES CUSTOMER(EMAIL) ON DELETE CASCADE INITIALLY DEFERRED, CONSTRAINT PO\_PK PRIMARY KEY(EMAIL));

CREATE TABLE SIGNS (EXISTING\_CUSTOMER\_EMAIL VARCHAR(255), CONTRACT\_ID INT , SUBDIVISION\_NAME VARCHAR(255));

ALTER TABLE SIGNS ADD CONSTRAINT SIGN\_EX\_FK FOREIGN KEY (EXISTING\_CUSTOMER\_EMAIL) REFERENCES EXISTING(EMAIL) ON DELETE CASCADE

INITIALLY DEFERRED;

ALTER TABLE SIGNS ADD CONSTRAINT SIGN\_CON\_FK FOREIGN KEY (CONTRACT\_ID) REFERENCES CONTRACT(ID) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE SIGNS ADD CONSTRAINT SIGN\_SUB\_FK FOREIGN KEY (SUBDIVISION\_NAME) REFERENCES SUBDIVISION(NAME) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE SIGNS ADD CONSTRAINT SIGN\_PK PRIMARY KEY(EXISTING\_CUSTOMER\_EMAIL, CONTRACT\_ID, SUBDIVISION\_NAME);

CREATE TABLE CUSTOMERPHONE (EMAIL VARCHAR(255) , PHONE VARCHAR(13), CONSTRAINT PHONE\_CUS\_FK FOREIGN KEY (EMAIL) REFERENCES CUSTOMER(EMAIL) ON DELETE CASCADE INITIALLY DEFERRED, CONSTRAINT CUS\_PH\_PK PRIMARY KEY (EMAIL,PHONE));

Alter table CUSTOMERPHONE add constraint "PHONE\_CHECK" CHECK (REGEXP\_LIKE(PHONE, '([0-9]{3})[0-9]{3}-[0-9]{4}'));

CREATE TABLE LOAN( ID INT , TERM INT , TYPE VARCHAR(255), GRANT\_DATE DATE, STATUS VARCHAR(20) CHECK(STATUS IN ('APPROVED','DISAPPROVED') ),LOANAPPLICANT\_ID INT);

ALTER TABLE LOAN ADD CONSTRAINT LOAN\_LA\_FK FOREIGN KEY (LOANAPPLICANT\_ID) REFERENCES LOANAPPLICANT(ID) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE LOAN ADD CONSTRAINT LOAN\_PK PRIMARY KEY(ID);

ALTER TABLE MANAGER ADD CONSTRAINT FK\_EMPLOYEE FOREIGN KEY (MANAGER\_ID) REFERENCES EMPLOYEE (ID) ON DELETE CASCADE INITIALLY DEFERRED;

ALTER TABLE CUSTOMER ADD DOB DATE ;

alter table CUSTOMER add constraint "EMAILFORMAT\_CHK" check ( REGEXP\_LIKE(EMAIL, '[a-zA-Z0-9.\_%~]+@[a-zA-Z0-9.\_%~]+\.[a-zA-Z]{2,4}'));

alter table EXISTING add constraint "EXISTINGEMAIL\_CHK" check ( REGEXP\_LIKE(EMAIL, '[a-zA-Z0-9.\_%~]+@[a-zA-Z0-9.\_%~]+\.[a-zA-Z]{2,4}'));

alter table CUSTOMERPHONE add constraint "CUSTOMERPHONE\_EMAIL\_CHK" check ( REGEXP\_LIKE(EMAIL, '[a-zA-Z0-9.\_%~]+@[a-zA-Z0-9.\_%~]+\.[a-zA-Z]{2,4}'));

alter table POTENTIAL add constraint "POTENTIAL\_EMAIL\_CHK" check ( REGEXP\_LIKE(EMAIL, '[a-zA-Z0-9.\_%~]+@[a-zA-Z0-9.\_%~]+\.[a-zA-Z]{2,4}'));

alter table VISITS add constraint "VISITS\_EMAIL\_CHK" check ( REGEXP\_LIKE(CUSTOMER\_EMAIL, '[a-zA-Z0-9.\_%~]+@[a-zA-Z0-9.\_%~]+\.[a-zA-Z]{2,4}'));

alter table SIGNS add constraint "SIGNS\_EMAIL\_CHK" check ( REGEXP\_LIKE(EXISTING\_CUSTOMER\_EMAIL, '[a-zA-Z0-9.\_%~]+@[a-zA-Z0-9.\_%~]+\.[a-zA-Z]{2,4}'));

### 3.1.3 A Database State

We insert some values into the database in order to test our SQL create view and query statement. Here we just give one example of insertions as follows:

#### INSERTION OF TABLE EMPLOYEE

```
INSERT INTO EMPLOYEE("ID","FNAME","LNAME","YEARS_OF_EXPERIENCE","E_TYPE","SALARY")
VALUES (1, 'Indhu', 'Surya', 5, 'SALES_AGENT', 25000);
```

#### EMPLOYEE

	ID	FNAME	LNAME	YEARS_OF_EXPERIENCE	E_TYPE	SALARY
1	1	Indhu	Surya	5	SALES_AGENT	25000
2	2	Sindhuja	R	15	SALES_AGENT	25345
3	3	Suneesha	Kudipudi	25	SALES_AGENT	35000
4	4	John	Smith	35	SALES_AGENT	25000
5	5	Mark	Twain	45	SALES_AGENT	25000
6	6	Sneha	Vishwa	50	SALES_AGENT	25000
7	7	Anisha	Chikerur	30	SALES_AGENT	25000
8	8	Sumithra	Shenoy	20	SALES_AGENT	25000
9	9	Swetha	Krishnakumar	5	SECRETARY	55500
10	10	Sathvi	Anish	5	SECRETARY	33000

#### INSERTION OF TABLE MANAGER

```
INSERT INTO MANAGER(MANAGER_ID, START_DATE) VALUES(1, TO_DATE('06051992','MMDDYYYY'));
```

#### MANAGER

	MANAGER_ID	START_DATE
1	1	05-JUN-92
2	2	05-JUL-92
3	3	14-AUG-92
4	4	05-DEC-92
5	5	20-SEP-92
6	6	10-JAN-00
7	7	15-FEB-02
8	8	06-JUN-15

#### INSERTION OF TABLE SUBDIVISION

```
INSERT INTO SUBDIVISION(NAME, START_YEAR, RANK, ZIP, MANAGER_ID) VALUES('SD1', 1993, 5,
75252, 2);
```

	NAME	START_YEAR	RANK	ZIP	MANAGER_ID
1	SD1	1993	5	75252	2
2	SD2	1993	5	75252	1
3	SD3	2015	5	75252	(null)
4	SD4	1993	7	75252	3
5	SD5	1993	8	75252	4
6	SD6	2015	2	75252	5
7	SD7	1993	1	75252	6
8	SD8	1993	11	75252	7
9	SD9	2015	10	75252	8

### INSERTION OF TABLE CUSTOMER

INSERT INTO CUSTOMER( EMAIL , NAME , INCOME, CREDIT\_SCORE, BATH\_COUNT, TYPE, BED\_COUNT, SQ\_FT, RANK, DOB ) VALUES ('ABC@GMAIL.COM' , 'CUST1' , 25000, 350, 1, 'EXISTING', 2, 2500, 5, TO\_DATE('06051992','MMDDYYYY') );

#### CUSTOMER

	EMAIL	NAME	INCOME	CREDIT_SCORE	BATH_COUNT	TYPE	BED_COUNT	SQ_FT	RANK	LOANAPPLICANT_ID	DOB
1	ABC@GMAIL.COM	CUST1	25000	350	1	EXISTING	2	2500	5	(null)	05-JUN-92
2	EFG@GMAIL.COM	CUST3	45000	500	1	EXISTING	3	4500	2	10003	05-NOV-92
3	IJK@YAHOO.COM	CUST4	25000	300	1	EXISTING	1	2500	3	10004	14-NOV-92
4	LMN@GMAIL.COM	CUST5	235000	400	1	EXISTING	2	3500	2	(null)	13-NOV-00
5	OPO@GMAIL.COM	CUST6	24000	450	2	EXISTING	2	4500	1	(null)	10-JAN-00
6	RST@GMAIL.COM	CUST7	252000	650	2	EXISTING	3	5500	1	(null)	28-NOV-83
7	UVW@GMAIL.COM	CUST8	28000	350	1	EXISTING	1	2500	3	(null)	28-NOV-83
8	CDE@GMAIL.COM	CUST2	35000	550	2	EXISTING	2	3500	4	(null)	20-SEP-92
9	XYZ@GMAIL.COM	CUST9	235000	750	1	POTENTIAL	2	3500	2	(null)	13-NOV-00
10	BCD@GMAIL.COM	CUST10	24000	750	2	POTENTIAL	2	4500	1	(null)	10-JAN-00
11	ZXY@GMAIL.COM	CUST11	252000	650	2	POTENTIAL	3	5500	1	(null)	28-NOV-83

### INSERTION OF TABLE LOT

INSERT INTO LOT(SUBDIVISION\_NAME, LOT\_NUMBER, LOT\_SIZE, FACING , AVAILABILITY ) VALUES('SD1', 1001, 500, 'NORTH', 'YES');

#### LOT

	SUBDIVISION_NAME	LOT_NUMBER	LOT_SIZE	FACING	AVAILABILITY
1	SD1	1001	500	NORTH	YES
2	SD2	1002	100	SOUTH	NO
3	SD3	1003	200	EAST	YES
4	SD4	1004	300	WEST	YES
5	SD1	1005	500	SOUTH	NO
6	SD2	1006	200	NORTH	YES
7	SD5	1007	100	NORTH	NO

### INSERTION OF TABLE FLOORPLAN

```
INSERT INTO FLOORPLAN(SUBDIVISION_NAME,FP_NUMBER , FLOOR_COUNT , BED_COUNT ,
BATH_COUNT , NAME , PRICE , SQ_FT ,GARAGE_COUNT) VALUES ('SD1', 101 , 1 , 3 , 2 , 'FP1' , 500 ,
3625 ,1);
```

### FLOORPLAN

	SUBDIVISION_NAME	FP_NUMBER	FLOOR_COUNT	BED_COUNT	BATH_COUNT	NAME	PRICE	SQ_FT	GARAGE_COUNT
1	SD1	101	1	3	2	FP1	500	3625	2
2	SD2	102	2	4	2	FP1	600	4625	2
3	SD3	103	3	2	2	FP2	700	5625	2
4	SD4	104	3	5	3	FP2	800	6625	3
5	SD5	105	2	4	2	FP1	600	4625	2
6	SD2	106	2	2	1	FP3	600	3625	1
7	SD1	107	1	1	1	FP4	500	2625	1

### INSERTION OF TABLE INVENTORY

```
INSERT INTO INVENTORY( SUBDIVISION_NAME , ID ,PRICE, MOVE_IN_DATE ) VALUES('SD1', 10000, 56,
TO_DATE('05091992', 'MMDDYYYY') );
```

### INVENTORY

	SUBDIVISION_NAME	ID	PRICE	MOVE_IN_DATE
1	SD1	10000	56	09-MAY-92
2	SD2	10001	5634	04-JUN-00
3	SD3	10002	5623	03-JUL-01
4	SD1	10003	5456	06-FEB-15
5	SD5	10004	5456	07-APR-15
6	SD3	10005	3454	(null)
7	SD2	10006	3456	08-NOV-15
8	SD4	10007	5236	09-DEC-14

### INSERTION OF TABLE CONTRACT

```
INSERT INTO CONTRACT(ID, SIGN_DATE , SALE_PRICE, EMPLOYEE_ID)
VALUES(1,TO_DATE('11282014','MMDDYYYY'),5000,1);
```

### CONTRACT

	ID	SIGN_DATE	SALE_PRICE	EMPLOYEE_ID
1	1	28-NOV-14	5000	1
2	2	24-NOV-14	4000	2
3	3	02-SEP-00	3000	3
4	4	04-APR-92	5000	1
5	5	04-FEB-12	3000	2
6	6	03-NOV-13	4000	1
7	7	15-NOV-14	2000	5
8	8	18-NOV-15	4000	5

INSERTION OF TABLE EXISTING

```
INSERT INTO EXISTING(EMAIL) VALUES ('ABC@GMAIL.COM');
```

EXISTING

	EMAIL
1	ABC@GMAIL.COM
2	CDE@GMAIL.COM
3	EFG@GMAIL.COM
4	IJK@YAHOO.COM
5	LMN@GMAIL.COM
6	OPQ@GMAIL.COM
7	RST@GMAIL.COM
8	UVW@GMAIL.COM

INSERTION OF TABLE POTENTIAL

```
INSERT INTO POTENTIAL(EMAIL) VALUES ('XYZ@GMAIL.COM');
```

POTENTIAL

	EMAIL
1	XYZ@GMAIL.COM
2	BCD@GMAIL.COM
3	ZXY@GMAIL.COM

INSERTION OF TABLE LOANAPPLICANT

```
INSERT INTO LOANAPPLICANT(ID) VALUES(10001);
```

LOANAPPLICANT

	ID
1	10001
2	10002
3	10003
4	10004
5	10005

INSERTION OF TABLE LOAN

```
INSERT INTO LOAN(ID, TERM, TYPE, GRANT_DATE, STATUS, LOANAPPLICANT_ID)
VALUES(101,7,'FIXED RATE', TO_DATE('11282014','MMDDYYYY'), 'APPROVED', 10001);
```

LOAN

	ID	TERM	TYPE	GRANT_DATE	STATUS	LOANAPPLICANT_ID
1	101	7	FIXED RATE	28-NOV-14	APPROVED	10001
2	102	15	ARM	02-SEP-00	DISAPPROVED	10002
3	103	7	FIXED RATE	28-NOV-15	APPROVED	10003
4	104	4	ARM	28-SEP-15	APPROVED	10004
5	105	5	FIXED RATE	28-OCT-00	APPROVED	10002

### INSERTION OF TABLE SECRETARY

INSERT INTO SECRETARY(EMPLOYEE\_ID, HOURLY\_WAGE) VALUES(9,30);

#### SECRETARY

	EMPLOYEE_ID	HOURLY_WAGE
1	9	30
2	10	35

### INSERTION OF TABLE SALES\_AGENT

INSERT INTO SALES\_AGENT(EMPLOYEE\_ID, SALES\_SHIFT) VALUES(1,'monday');

#### SALES\_AGENT

	EMPLOYEE_ID	SALES_SHIFT
1	1	monday
2	2	tuesday
3	4	monday
4	5	monday
5	6	wednesday
6	7	monday
7	1	friday
8	2	saturday
9	3	sunday
10	4	wednesday
11	1	thrusday
12	3	wednesday

### INSERTION OF TABLE CUSTOMERPHONE

INSERT INTO CUSTOMERPHONE(EMAIL,PHONE) VALUES('ABC@GMAIL.COM', '(999)999-9999');

#### CUSTOMERPHONE



	EMAIL	PHONE
1	ABC@GMAIL.COM	(999) 999-9999
2	BCD@GMAIL.COM	(999) 999-9999
3	CDE@GMAIL.COM	(999) 999-9999
4	EFG@GMAIL.COM	(999) 999-9999
5	IJK@YAHOO.COM	(999) 999-9999
6	LMN@GMAIL.COM	(999) 999-9999
7	OPQ@GMAIL.COM	(999) 999-9999
8	RST@GMAIL.COM	(999) 999-9999
9	UVW@GMAIL.COM	(999) 999-9999
10	ABC@GMAIL.COM	(349) 345-9999
11	UVW@GMAIL.COM	(943) 943-9999
12	ABC@GMAIL.COM	(439) 945-9999
13	EFG@GMAIL.COM	(932) 943-9349
14	IJK@YAHOO.COM	(949) 349-9349

### INSERTION OF TABLE EMPLOYEE\_PHONE

```
INSERT INTO EMPLOYEEPHONE(EMPLOYEE_ID,PHONE) VALUES(1, '(999)999-9999');
```

### EMPLOYEEPHONE

	EMPLOYEE_ID	PHONE
1	1	(999) 999-9999
2	2	(999) 999-9999
3	3	(999) 999-9999
4	4	(999) 999-9999
5	7	(999) 999-9999
6	10	(999) 999-9999
7	1	(456) 999-9999
8	1	(753) 999-9999
9	2	(432) 999-9999
10	3	(865) 999-9999
11	6	(437) 999-9999

### INSERTION OF TABLE VISITS

```
INSERT INTO VISITS(CUSTOMER_EMAIL, SUBDIVISION_NAME, VISIT_DATE)
VALUES('ABC@GMAIL.COM','SD1',TO_DATE('05092000', 'MMDDYYYY'));
```

### VISITS

	CUSTOMER_EMAIL	SUBDIVISION_NAME	VISIT_DATE
1	ABC@GMAIL.COM	SD1	09-MAY-00
2	ABC@GMAIL.COM	SD2	09-JUN-01
3	ABC@GMAIL.COM	SD3	09-NOV-14
4	ABC@GMAIL.COM	SD4	09-DEC-15
5	CDE@GMAIL.COM	SD1	09-SEP-15
6	OPQ@GMAIL.COM	SD1	09-AUG-11
7	CDE@GMAIL.COM	SD3	09-JUL-00
8	UVW@GMAIL.COM	SD4	09-JUN-15
9	UVW@GMAIL.COM	SD5	09-APR-14
10	EFG@GMAIL.COM	SD2	09-MAY-93
11	EFG@GMAIL.COM	SD6	09-MAR-92
12	IJK@YAHOO.COM	SD2	09-MAY-92

### INSERTION OF TABLE SIGNS

```
INSERT INTO SIGNS(EXISTING_CUSTOMER_EMAIL, CONTRACT_ID, SUBDIVISION_NAME)
VALUES('ABC@GMAIL.COM', 1, 'SD1');
```

#### SIGNS

	EXISTING_CUSTOMER_EMAIL	CONTRACT_ID	SUBDIVISION_NAME
1	ABC@GMAIL.COM	1	SD1
2	ABC@GMAIL.COM	2	SD4
3	ABC@GMAIL.COM	3	SD3
4	CDE@GMAIL.COM	4	SD2
5	EFG@GMAIL.COM	5	SD5
6	EFG@GMAIL.COM	6	SD4
7	OPQ@GMAIL.COM	7	SD3
8	OPQ@GMAIL.COM	8	SD2

### INSERTION OF TABLE ASSOCIATES

```
INSERT INTO ASSOCIATES(L_SUBDIVISION_NAME, LOT_NUMBER, F_SUBDIVISION_NAME,
FLOORPLAN_NUMBER, CONTRACT_ID) VALUES('SD1', 1001, 'SD1', 101, 1);
```

#### ASSOCIATES

	I_SUBDIVISION_NAME	LOT_NUMBER	F_SUBDIVISION_NAME	FLOORPLAN_NUMBER	CONTRACT_ID
1	SD1	1001	SD1	101	1
2	SD1	1001	SD2	102	2
3	SD1	1005	SD1	101	3
4	SD1	1005	SD2	102	4
5	SD2	1002	SD1	101	5
6	SD2	1006	SD2	102	6
7	SD4	1004	SD2	106	8
8	SD5	1007	SD2	106	1
9	SD3	1003	SD2	106	7

### INSERTION OF TABLE TOURS

INSERT INTO TOURS(EMP\_ID, CUSTOMER\_EMAIL, I\_SUBDIVISION\_NAME, INVENTORY\_ID, TOUR\_DATE) VALUES(1, 'ABC@GMAIL.COM', 'SD1', 10000, TO\_DATE('05092000', 'MMDDYYYY'));

#### TOURS

	EMP_ID	CUSTOMER_EMAIL	I_SUBDIVISION_NAME	INVENTORY_ID	TOUR_DATE
1	1	ABC@GMAIL.COM	SD1	10000	09-MAY-00
2	3	ABC@GMAIL.COM	SD1	10000	29-DEC-04
3	2	ABC@GMAIL.COM	SD1	10000	19-NOV-03
4	1	IJK@YAHOO.COM	SD4	10007	09-AUG-05
5	3	IJK@YAHOO.COM	SD4	10007	05-JUL-07
6	2	LMN@GMAIL.COM	SD5	10004	03-MAY-08
7	4	IJK@YAHOO.COM	SD5	10004	02-MAR-06
8	3	RST@GMAIL.COM	SD3	10005	01-MAR-15
9	2	RST@GMAIL.COM	SD2	10001	18-APR-14

### INSERTION OF TABLE COME

INSERT INTO COME(I\_SUBDIVISION\_NAME, INVENTORY\_ID, L\_SUBDIVISION\_NAME, LOT\_NUMBER, F\_SUBDIVISION\_NAME, FLOORPLAN\_NUMBER) VALUES('SD1', 10000, 'SD1', 1001, 'SD3', 103);

#### COME

	I_SUBDIVISION_NAME	INVENTORY_ID	L_SUBDIVISION_NAME	LOT_NUMBER	F_SUBDIVISION_NAME	FLOORPLAN_NUMBER
1	SD1	10000	SD1	1001	SD3	103
2	SD1	10000	SD1	1005	SD4	104
3	SD2	10001	SD2	1002	SD3	103
4	SD2	10001	SD2	1006	SD1	107
5	SD3	10002	SD3	1003	SD2	102
6	SD3	10002	SD2	1002	SD1	101
7	SD3	10005	SD3	1003	SD3	103
8	SD3	10005	SD2	1002	SD1	107
9	SD5	10004	SD1	1005	SD3	103
10	SD5	10004	SD3	1003	SD4	104

UPDATE OF TABLE EMPLOYEE

UPDATE EMPLOYEE SET SUBDIVISION\_NAME='SD1', LOANAPPLICANT\_ID='10001' WHERE ID=1;

UPDATE EMPLOYEE SET SUBDIVISION\_NAME='SD1' WHERE ID=2;

EMPLOYEE

ID	FNAME	LNAME	YEARS_OF_EXPERIENCE	E_TYPE	SALARY	SUBDIVISION_NAME	LOANAPPLICANT_ID
1	1 Indhu	Surya		5 SALES_AGENT	25000	SD1	10001
2	2 Sindhuja	R		15 SALES_AGENT	25345	SD1	(null)
3	3 Suneesha	Kudipudi		25 SALES_AGENT	35000	SD2	10002
4	4 John	Smith		35 SALES_AGENT	25000	SD1	(null)
5	5 Mark	Twain		45 SALES_AGENT	25000	SD7	10005
6	6 Sneha	Vishwa		50 SALES_AGENT	25000	SD2	(null)
7	7 Anisha	Chikerur		30 SALES_AGENT	25000	SD3	(null)
8	8 Sumithra	Shenoy		20 SALES_AGENT	25000	SD3	(null)
9	9 Swetha	Krishnakumar		5 SECRETARY	55500	SD5	(null)
10	10 Sathvi	Anish		5 SECRETARY	33000	SD6	(null)

Till now we finished the process of creating tables and database states.

### 3.2 Creation of Views (Answer for Question d)

Use the Create View statement to create the following views:

1. School rank: show the school rank of each subdivision along with the average base price of floor plans the subdivision provides.

Create view school\_rank as select s.name as subdivisionname, rank, avg(f.price) as avg\_base\_price from subdivision s join floorplan f on s.name = f.subdivision\_name group by s.name, rank;

SELECT \* FROM SCHOOL\_RANK;

	SUBDIVISIONNAME	RANK	AVG_BASE_PRICE
1	SD3	5	700
2	SD4	7	800
3	SD2	5	600
4	SD1	5	500
5	SD5	8	600

2. Promising customers: show the name, age and email address of potential customers with credit score over 740.

Create view promising\_customers as select name, floor(months\_between(SYSDATE, DOB) /12) as age, p.email from potential p join customer c on p.email = c.email where credit\_score > 740;

SELECT \* FROM PROMISING\_CUSTOMERS;

	NAME	AGE	EMAIL
1	CUST9	15	XYZ@GMAIL.COM
2	CUST10	15	BCD@GMAIL.COM

3. Inventory homes: show the information of unsold inventory homes that have been completed or will be completed by the end of 2015 for each subdivision along with the manager name of the subdivision.

Create view **inventory\_homes** as select i.subdivision\_name, concat(e.fname, e.lname) as manager, i.id as inventory\_id, price, move\_in\_date from inventory i join subdivision s on i.subdivision\_name = s.name join employee e on s.manager\_id = e.id where move\_in\_date is null group by i.subdivision\_name, concat(e.fname, e.lname), i.id, price, move\_in\_date;

SELECT \* FROM INVENTORY\_HOMES;

	SUBDIVISION_NAME	MANAGER	INVENTORY_ID	PRICE	MOVE_IN_DATE
1	SD3	SwethaKrishnakumar	10005	3454	(null)

4. Large floor plans: show the information of floor plans over 4000 square feet for each subdivision.

Create view **Large\_floor\_plans** as Select subdivision\_name, fp\_number, floor\_count, bed\_count, bath\_count, name, price, sq\_ft, garage\_count from floorplan where sq\_ft > 4000 group by subdivision\_name, fp\_number, floor\_count, bed\_count, bath\_count, name, price, sq\_ft, garage\_count;

SELECT \* FROM LARGE\_FLOOR\_PLANS;

	SUBDIVISION_NAME	FP_NUMBER	FLOOR_COUNT	BED_COUNT	BATH_COUNT	NAME	PRICE	SQ_FT	GARAGE_COUNT
1	SD4	104	3	5	3	FP2	800	6625	3
2	SD5	105	2	4	2	FP1	600	4625	2
3	SD3	103	3	2	2	FP2	700	5625	2
4	SD2	102	2	4	2	FP1	600	4625	2

5. Sales record: for each subdivision, show the name of each sales agent and the number of contracts she prepared in year 2014.

Create view **Sales\_Record** as Select subdivision\_name, e.id as employee\_id, concat(fname, lname) as name, count(c.id) as number\_of\_contracts from employee e join contract c on e.id = c.employee\_id where to\_char(sign\_date,'yyyy') = '2014' group by subdivision\_name, e.id, concat(fname, lname);

SELECT \* FROM SALES\_RECORD;

	SUBDIVISION_NAME	EMPLOYEE_ID	NAME	NUMBER_OF_CONTRACTS
1	SD1	1	IndhuSurya	1
2	SD1	2	SindhujaR	1
3	SD7	5	MarkTwain	1

6. Deal discount: for each subdivision, show the information of the contracts signed for inventory homes with a discount in year 2015. Here contract with discount means that the sales price showed in the contract is lower than the list price of the inventory home.

Create view **Deal\_discount** as select sign.subdivision\_name, c.id as contracted , sign\_date, sale\_price, employee\_id from contract c join signs sign on c.id = sign.contract\_id join associates a on sign.contract\_id = a.contract\_id join come c1 on (a.l\_subdivision\_name = c1.l\_subdivision\_name and a.lot\_number = c1.lot\_number and a.f\_subdivision\_name = c1.f\_subdivision\_name and a.floorplan\_number = c1.floorplan\_number) join inventory i on (c1.i\_subdivision\_name = i.subdivision\_name and c1.inventory\_id = i.id) where c.sale\_price < i.price and sign\_date >= to\_date('01/JAN/2015', 'dd/mon/yyyy') group by sign.subdivision\_name, c.id, sign\_date, sale\_price, employee\_id;

SELECT \* FROM DEAL\_DISCOUNT;

	SUBDIVISION_NAME	CONTRACTED	SIGN_DATE	SALE_PRICE	EMPLOYEE_ID
1	SD2	8	18-NOV-15	4000	5

### 3.3 Creation of SQL Queries (Answer for Question f)

Now we give out the SQL Queries for each of 10 questions listed in Question e as follows:

1. Retrieve the school rank of each subdivision in decreasing order of the average base price of floor plans the subdivision provides.

**Select \* from school\_rank order by avg\_base\_price desc;**

	SUBDIVISIONNAME	RANK	AVG_BASE_PRICE
1	SD4	7	800
2	SD3	5	700
3	SD5	8	600
4	SD2	5	600
5	SD1	5	500

2. For each subdivision, retrieve the number of unsold inventory homes that have been completed or will be ready by the end of year 2015.

**Select subdivision\_name, count(inventory\_id) as unsold\_inventory from inventory\_homes group by subdivision\_name;**

	SUBDIVISION_NAME	UNSOLD_INVENTORY
1	SD3	1

3. For each subdivision, retrieve information of the sales agents who prepared all the contracts with floor plans over 4000 square feet signed in year 2015

**Select lfp.subdivision\_name, c.employee\_id, c.id from Large\_floor\_plans lfp join associates a on (lfp.subdivision\_name = a.f\_subdivision\_name and lfp.fp\_number = a.floorplan\_number) join contract c on a.contract\_id = c.id where c.employee\_id in (select employee\_id from SALES\_AGENT) group by lfp.subdivision\_name, c.employee\_id, c.id;**

	SUBDIVISION_NAME	EMPLOYEE_ID	ID
1	SD2	1	6
2	SD2	1	4
3	SD2	2	2

4. For each subdivision, retrieve the information of the sales agent who prepared the highest number of contracts in year 2014.

**Select \* from employee e join sales\_record sr on e.id = sr.employee\_id where sr.number\_of\_contracts = (select max(number\_of\_contracts) from sales\_record);**

ID	FNAME	LNAME	YEARS...	E_TYPE	SALARY	SUBDIVISION_N...	LOANAPPLICA...	SUBDIVISION_...	EMPLOYEE_ID	NAME	NUMBER_OF_CONTRACTS
1	Indhu	Surya	5	SALES_AGENT	25000	SD1		10001 SD1		1 IndhuSurya	1
2	Sindhuja	R	15	SALES_AGENT	25345	SD1		(null) SD1		2 SindhujaR	1
5	Mark	Twain	45	SALES_AGENT	25000	SD7		10005 SD7		5 MarkTwain	1

5. Retrieve the information of the inventory home that has not been sold for the longest time since its completion, along with the name of the manager who is in charge of that subdivision.

**Select \* from (select \* from inventory\_homes order by move\_in\_date) where rownum = 1;**

	SUBDIVISION_NAME	MANAGER	INVENTORY_ID	...	MOVE_IN_DATE
1	SD3	SwethaKrishnakumar	10005	3454	(null)

6. Retrieve the information of potential customers who have been visiting subdivisions of the company for more than one month and the price range of the inventory homes they have toured lies between \$300,000 to \$400,000.

**Select c.email, c.name, c.income, c.credit\_score, i.price from CUSTOMER c join potential p on c.email = p.email join tours t on p.email = t.customer\_email join inventory i on (t.i\_subdivision\_name = i.subdivision\_name and t.inventory\_id = i.id) where i.price between 300000 and 400000;**

	EMAIL	NAME	INCOME	CREDIT_SCORE	PRICE
1	XYZ@GMAIL.COM	CUST9	235000	750	350000
2	BCD@GMAIL.COM	CUST10	24000	750	350000

7. For each subdivision, retrieve the information of each customer whose home loan has not been approved over one month after he/she signed his/her contract.

**Select sign.subdivision\_name, sign.existing\_customer\_email, cust.name, c.id as contract\_id, c.sign\_date, sale\_price, status from contract c join signs sign on c.id = sign.contract\_id join customer cust on sign.existing\_customer\_email = cust.email join loan on cust.loanapplicant\_id = loan.loanapplicant\_id where (status is null or status = 'DISAPPROVED') and months\_between(sysdate, sign\_date) > 1 group by sign.subdivision\_name, sign.existing\_customer\_email, cust.name, c.id, c.sign\_date, sale\_price, status;**

	SUBDIVISION_...	EXISTING_CUSTOMER_EMAIL	NAME	CONTRACT_ID	SIGN_DATE	SALE_PRICE	STATUS
1	SD5	EFG@GMAIL.COM	CUST3	5	04-FEB-12	3000	(null)
2	SD4	EFG@GMAIL.COM	CUST3	6	03-NOV-13	4000	(null)


8. Retrieve the name, age and email address of potential customers with credit score over 740 who visited a subdivision in August this year but have not visited any subdivision since September 1st .

**Select \* from promising\_customers pc, visits v where pc.email = v.customer\_email and upper(to\_char(v.visit\_date, 'Month')) = 'AUGUST' and not exists (select 1 from visits vs where vs.customer\_email = pc.email and visit\_date >= to\_date('01-SEP-2015', 'dd-mon-yy'));**

NAME	AGE	EMAIL	CUSTOMER...	SUBDIVIS...	VISIT_DATE
------	-----	-------	-------------	-------------	------------

9. Retrieve the average discount received by the customers who have signed a contract to purchase an inventory home.

**Select avg(price-sale\_price) from contract c join signs sign on c.id = sign.contract\_id join associates a on sign.contract\_id = a.contract\_id join come c1 on (a.l\_subdivision\_name = c1.l\_subdivision\_name and a.lot\_number = c1.lot\_number and a.f\_subdivision\_name = c1.f\_subdivision\_name and a.floorplan\_number = c1.floorplan\_number) join inventory i on (c1.i\_subdivision\_name = i.subdivision\_name and c1.inventory\_id = i.id) where c.sale\_price < i.price;**

	 AVG(PRICE-SALE_PRICE)
1	2123

10. Retrieve information of the potential customers whose house preferences can be matched by one or more inventory homes that will be ready to move in by the end of year 2015

```
select customer.email, customer.name from inventory i join come c on (c.i_subdivision_name = i.subdivision_name and c.inventory_id = i.id) join floorplan fp on (c.f_subdivision_name = fp.subdivision_name and c.floorplan_number = fp.fp_number) , customer where customer.bath_count=fp.bath_count and customer.bed_count=fp.bed_count and customer.sq_ft=fp.sq_ft and upper(to_char(i.move_in_date,'Year')) = '2015' and customer.email in (select email from potential);
```

 EMAIL	 NAME
---	--

## 4. Conclusion

In this report we modified the EER diagram and relational schemas for XXX Database according to the requirement of Phase III. We also give dependency diagram for each relational schema in database. Then we created tables for each relational schema and write the SQL statements for the views and queries listed in Question d and Question e.