**TABLES**

CREATE DATABASE HomeSync;

USE HomeSync

CREATE TABLE Room(

room\_id INT PRIMARY KEY IDENTITY,

type varchar(20),

floor INT,

status varchar(20)

);

CREATE TABLE Users (

id int PRIMARY KEY IDENTITY,

f\_Name varchar (20),

l\_Name varchar (20),

password varchar (10),

email varchar (50),

preference varchar (100),

room int,

type varchar(20),

birthdate DATETIME not null,

age AS (YEAR(CURRENT\_TIMESTAMP)-YEAR(birthdate)),

FOREIGN KEY(room) REFERENCES Room ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Admin (

admin\_id int,

no\_of\_guests\_allowed int,

salary decimal (13,2),

PRIMARY KEY (admin\_id),

FOREIGN KEY(admin\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Guest (

guest\_id int,

guest\_of int,

address varchar (30),

arrival\_date DATETIME,

departure\_date DATETIME,

residential bit,

PRIMARY KEY (guest\_id),

FOREIGN KEY(guest\_id) REFERENCES Users ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(guest\_of) REFERENCES Admin

);

--------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------

CREATE TABLE Task (

Task\_id int PRIMARY KEY IDENTITY,

name varchar(30),

creation\_date DATETIME,

due\_date DATETIME,

category varchar(20),

creator int,

status varchar(20),

reminder\_date DATETIME,

priority int,

FOREIGN KEY(creator) REFERENCES Admin ON DELETE CASCADE ON UPDATE CASCADE

);

--------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------

CREATE TABLE Assigned\_to(

admin\_id int,

task\_id int,

user\_id int,

PRIMARY KEY(admin\_id, task\_id, user\_id),

FOREIGN KEY(admin\_id) REFERENCES Admin,

FOREIGN KEY(user\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(task\_id) REFERENCES Task

);

--------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------

CREATE TABLE Calendar(

event\_id int ,

user\_assigned\_to int,

name varchar(50),

description varchar (200),

location varchar (40),

reminder\_date DATETIME,

PRIMARY KEY(event\_id, user\_assigned\_to),

FOREIGN KEY(user\_assigned\_to) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE

);

-------------------------------Malak---------------------------------------------------------------------------------

CREATE TABLE Notes(

id INT,

user\_id INT,

content varchar(500),

creation\_date DATETIME,

title varchar(50),

PRIMARY KEY(id),

FOREIGN KEY(user\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE,

);

-- USER

CREATE TABLE Travel(

trip\_no INT,

hotel\_name varchar(10),

destination varchar(40),

ingoing\_flight\_num varchar(30),

outgoing\_flight\_num varchar(30),

ingoing\_flight\_date DATETIME,

outgoing\_flight\_date DATETIME,

ingoing\_flight\_airport varchar(30),

outgoing\_flight\_airport varchar(30),

transport varchar(10),

PRIMARY KEY(trip\_no)

);

CREATE TABLE User\_trip(

trip\_no INT,

user\_id INT,

hotel\_room\_no INT,

in\_going\_flight\_seat\_number VARCHAR(5),

out\_going\_flight\_seat\_number VARCHAR(5),

PRIMARY KEY(trip\_no,user\_id),

FOREIGN KEY(trip\_no) REFERENCES Travel ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(user\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE,

);

-- USER AND TRIP

CREATE TABLE Finance(

payement\_id INT IDENTITY,

user\_id INT,

type varchar(30),

amount DECIMAL(13,2),

currency varchar(3),

method varchar(10),

status varchar(10),

date DATETIME,

receipt\_no INT,

deadline DATETIME,

penalty DECIMAL,

PRIMARY KEY(payement\_id),

FOREIGN KEY(user\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE,

);

-- USER

CREATE TABLE Health(

user\_id INT,

date DATETIME,

activity varchar(10),

hours\_slept INT,

food varchar(10),

PRIMARY KEY(date, activity),

FOREIGN KEY(user\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE,

);

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

CREATE TABLE Communication(

message\_id INT PRIMARY KEY IDENTITY,

sender\_id int,

reciever\_id int,

content varchar(200),

time\_sent DATETIME,

time\_received DATETIME,

time\_read DATETIME,

title varchar(30)

FOREIGN KEY(sender\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(reciever\_id) REFERENCES Users

);

CREATE TABLE Device(

device\_id INT PRIMARY KEY,

room INT,

type varchar(20),

status varchar(20),

battery\_status INT,

FOREIGN KEY(room) REFERENCES Room ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE RoomSchedule(

creator\_id INT ,

action varchar(20),

room INT,

start\_time DATETIME ,

end\_time DATETIME,

PRIMARY KEY(creator\_id, start\_time),

FOREIGN KEY(creator\_id) REFERENCES Users,

FOREIGN KEY(room) REFERENCES Room ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Log(

room\_id INT,

device\_id INT,

user\_id INT,

activity varchar(30),

date DATETIME,

duration varchar(50),

PRIMARY KEY(room\_id, device\_id, user\_id,date),

FOREIGN KEY(room\_id) REFERENCES Room ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(device\_id) REFERENCES Device,

FOREIGN KEY(user\_id) REFERENCES Users

);

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

USE HomeSync;

CREATE TABLE Consumption(

date DATETIME,

device\_id INT,

consumption INT,

PRIMARY KEY(device\_id , date),

FOREIGN KEY(device\_id) REFERENCES Device ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Preferences(

user\_id INT,

preference\_no INT,

category varchar(20),

content varchar(20),

PRIMARY KEY(user\_id ,preference\_no),

FOREIGN KEY(user\_id) REFERENCES Users ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Recommendation(

recommendation\_id INT PRIMARY KEY IDENTITY,

user\_id INT,

category varchar(20),

preference\_no INT,

content varchar(20),

FOREIGN KEY(user\_id, preference\_no) REFERENCES Preferences ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(user\_id) REFERENCES Users

);

CREATE TABLE Inventory(

supply\_id INT PRIMARY KEY ,

name varchar(30),

quantity INT,

expiry\_date DATETIME,

price decimal(10,2),

manufacturer varchar(30),

category varchar(20),

);

CREATE TABLE Camera(

monitor\_id INT,

camera\_id INT,

room\_id INT,

PRIMARY KEY(monitor\_id),

FOREIGN KEY(room\_id) REFERENCES Room ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (monitor\_id) REFERENCES Users

);

PROCEDURES

use HomeSync;

go

/\* 1.1 Sarah

Register on the website with a unique email along with the needed information. Choose which type

of user you will be using @usertype (Admin).

Signature:

Name: UserRegister.

Input: @usertype varchar(20), @userName varchar(20), @email varchar(50), @first\_name varchar(20),

@last\_name varchar(20), @birth\_date datetime, @password varchar(10).

Output: user\_id int.

\*/

CREATE PROCEDURE UserRegister

@usertype varchar(20),

@email varchar(50),

@first\_name varchar(20),

@last\_name varchar(20),

@birth\_date datetime,

@password varchar(10),

@user\_id int OUTPUT

AS

BEGIN

IF NOT EXISTS (SELECT 1 FROM Users WHERE email = @email) AND @usertype ='Admin'

BEGIN

INSERT INTO Users (f\_Name,l\_Name,password,email, birthdate, type)

VALUES (@first\_name , @last\_name, @password, @email, @birth\_date, @usertype);

SELECT @user\_id = id

FROM Users

WHERE f\_Name=@first\_Name AND l\_Name=@last\_name

AND email=@email AND password=@password

INSERT INTO Admin (admin\_id, no\_of\_guests\_allowed) values (@user\_id, 30);

END

ELSE IF NOT EXISTS (SELECT 1 FROM Users WHERE email = @email) AND @usertype ='Guest'

BEGIN

INSERT INTO Users (f\_Name,l\_Name,password,email, birthdate, type)

VALUES (@first\_name , @last\_name, @password, @email, @birth\_date, @usertype);

SELECT @user\_id = id

FROM Users

WHERE f\_Name=@first\_Name AND l\_Name=@last\_Name

AND email=@email AND password=@password

INSERT INTO Guest (guest\_id) values (@user\_id);

END

ELSE

SET @user\_id = -1;

END;

GO;

/\* 2.1 Sarah

Login using my email and password. In the event that the user is not registered, the @user\_id value

will be (-1).

Signature:

Name: UserLogin.

Input: @email varchar(50), @password varchar(10).

Output: @success bit , @user\_id int

\*/

CREATE PROCEDURE UserLogin

@email varchar(50), @password varchar(10),

@success bit OUTPUT, @user\_id int OUTPUT

AS

BEGIN

IF EXISTS (SELECT 1 FROM Users WHERE email = @email AND password = @password)

BEGIN

SET @success = 1

SELECT @user\_id = id

FROM Users

WHERE email = @email AND password = @password

END

ELSE

BEGIN

SET @success=0

SET @user\_id = -1;

END

END;

GO

/\* 2.2 Toqa

View all the details of my profile.

Signature:

Name: ViewProfile.

Input: @user\_id int.

Output: Table containing the user details.

\*/

CREATE PROCEDURE ViewProfile

@user\_id int

AS

BEGIN

SELECT \*

FROM Users

WHERE ID = @user\_id;

END;

GO;

/\* 2.4 Sarah

View their task. (You should check if the deadline has passed or not if it passed set the status to

done).

Signature:

Name: ViewMyTask.

Input: @user\_id int.

Output: Table containing all the details of the tasks

\*/

CREATE PROCEDURE ViewMyTask

@user\_id int

AS

BEGIN

UPDATE Task

SET status = 'Done'

WHERE creator = @user\_id AND due\_date < CURRENT\_TIMESTAMP;

SELECT \*

FROM Task

WHERE creator = @user\_id

END;

GO;

/\* 2.5 Sarah

Finish their task.

Signature:

Name: FinishMyTask.

Input: @user\_id int, @title varchar(50).

Output: Nothing.

\*/

CREATE PROCEDURE FinishMyTask

@user\_id int,

@title varchar(50)

AS

BEGIN

UPDATE Task

SET status = 'Done'

WHERE creator = @user\_id AND name=@title ;

END;

GO;

/\* 2.6 Sarah

View task status given the @user\_id and the @creator of the task. (The recently created reports

should be shown first).

Signature:

Name: ViewTask.

Input: @user\_id int, @creator int.

Output: Table containing all details of the task(s).

\*/

CREATE PROCEDURE ViewTask

@user\_id int,

@creator int

AS

BEGIN

SELECT t.\*

FROM Assigned\_to a INNER JOIN Task t on a.task\_id=t.Task\_id

WHERE a.admin\_id=@creator AND a.user\_id=@user\_id

ORDER BY t.creation\_date DESC;

END;

GO;

/\* 2.7 Phebe

View device charge.

Signature:

Name: ViewMyDeviceCharge.

Input: @device\_id int.

Output: @charge int, @location int.

\*/

CREATE PROCEDURE ViewMyDeviceCharge

@device\_id int,

@charge int OUTPUT,

@location int OUTPUT

AS

BEGIN

SELECT @charge = battery\_status ,@location= room

FROM Device

WHERE device\_id = @device\_id

END;

GO;

/\* 2.8 Phebe

Book a room with other users.

Signature:

Name: AssignRoom.

Input: @user\_id int,@room\_id int.

Output: Nothing.

\*/

CREATE PROCEDURE AssignRoom

@user\_id INT,

@room\_id INT

AS

BEGIN

IF EXISTS (SELECT 1 FROM Room WHERE room\_id = @room\_id AND status <> 'Booked')

AND EXISTS (SELECT 1 FROM Users WHERE id = @user\_id )

BEGIN

UPDATE Users

SET room = @room\_id

WHERE id = @user\_id;

UPDATE Room

SET status = 'Booked'

WHERE room\_id = @room\_id;

END;

ELSE

print 'room booked'

END;

GO;

/\* 2.9 Sarah

Create events on the system.

Signature:

Name: CreateEvent.

Input:@event\_id Int @user\_id int, @name varchar(50), @description varchar(200), @location varchar(40),@reminder\_date datetime ,@other\_user\_id int.

Output: Nothing

\*/

CREATE PROCEDURE CreateEvent

@event\_id int,

@user\_id int,

@name varchar(50),

@description varchar(200),

@location varchar(40),

@reminder\_date datetime,

@other\_user\_id int

AS

BEGIN

IF EXISTS (select 1 from Calendar where name=@name AND event\_id=@event\_id

AND user\_assigned\_to=@user\_id AND description=@description

AND location=@location AND reminder\_date=@reminder\_date)

print 'event exists already'

ELSE

BEGIN

INSERT INTO Calendar (event\_id, user\_assigned\_to, name, description, location, reminder\_date)

VALUES (@event\_id, @user\_id, @name, @description, @location, @reminder\_date);

IF @other\_user\_id IS NOT NULL

INSERT INTO Calendar (event\_id, user\_assigned\_to, name, description, location, reminder\_date)

VALUES (@event\_id, @other\_user\_id, @name, @description, @location, @reminder\_date);

END

END

GO;

/\* 2.10 Sarah

Assign user to attend event.

Signature:

Name: AssignUser.

Input: @user\_id int, @event\_id int.

Output: @user\_id and all details of the event that the employee is assigned to.

\*/

CREATE PROCEDURE AssignUser

@user\_id int,

@event\_id int

AS

BEGIN

IF EXISTS (SELECT 1 FROM Calendar WHERE event\_id = @event\_id)

BEGIN

if exists(SELECT 1 FROM Calendar WHERE user\_assigned\_to=@user\_id)

PRINT 'User already assigned to event'

ELSE

begin

DECLARE @name varchar(50);

DECLARE @description varchar(200);

DECLARE @location varchar(40);

DECLARE @reminder\_date datetime;

SELECT @name = name, @description=description, @location=location, @reminder\_date=reminder\_date

FROM Calendar

WHERE event\_id=@event\_id

INSERT INTO Calendar (event\_id, user\_assigned\_to, name, description, location, reminder\_date)

VALUES (@event\_id, @user\_id, @name, @description, @location, @reminder\_date);

SELECT \*

FROM Calendar

WHERE event\_id=@event\_id AND user\_assigned\_to=@user\_id;

end

END;

ELSE

PRINT 'Event does not exist';

END;

GO

/\* 2.11 Sarah

Add a reminder to a task.

Signature:

Name: AddReminder.

Input: @task\_id int, @reminder datetime.

Output: Nothing.

\*/

CREATE PROCEDURE AddReminder

@task\_id int,

@reminder datetime

AS

BEGIN

IF NOT EXISTS (select 1 from Task where Task\_id=@task\_id)

print 'task does not exist'

ELSE

BEGIN

UPDATE Task

SET reminder\_date = @reminder

WHERE Task\_id = @task\_id;

END

END

GO;

/\* 2.12 Sarah

Uninvite a specific user to an event.

Signature:

Name: Uninvited.

Input: @event\_id int, @user\_id int.

Output: Nothing.

\*/

CREATE PROCEDURE Uninvited

@event\_id int,

@user\_id int

AS

BEGIN

IF NOT EXISTS (select 1 from Calendar where event\_id=@event\_id AND user\_assigned\_to=@user\_id)

print 'Event does not exist'

ELSE

DELETE

FROM Calendar

WHERE event\_id=@event\_id AND user\_assigned\_to=@user\_id;

END;

GO;

/\* 2.13 Sarah

Update the deadline of a specific task.

Signature:

Name: UpdateTaskDeadline.

Input: @deadline datetime, @task\_id int.

Output: Nothing.

\*/

CREATE PROCEDURE UpdateTaskDeadline

@deadline datetime,

@task\_id int

AS

BEGIN

IF NOT EXISTS (select 1 from Task where Task\_id=@task\_id )

PRINT 'task does not exist'

ELSE

UPDATE Task

SET due\_date = @deadline

WHERE Task\_id=@task\_id

END;

GO;

/\* 2.14 Sarah

View their event given the @event\_id and if the @event\_id is empty then view all events that

belong to the user order by their date.

Signature:

Name: ViewEvent.

Input: @User\_id int, @Event\_id int.

Output: Table containing all the events details

\*/

CREATE PROCEDURE ViewEvent

@User\_id int,

@Event\_id int

AS

BEGIN

IF EXISTS (SELECT 1 FROM Calendar WHERE event\_id = @Event\_id AND user\_assigned\_to=@User\_id)

SELECT \*

FROM Calendar

WHERE event\_id = @Event\_id AND user\_assigned\_to=@User\_id;

ELSE

SELECT \*

FROM Calendar

WHERE user\_assigned\_to=@User\_id

ORDER BY reminder\_date;

END;

GO;

/\* 2.15 Toqa

View users that have no recommendations.

Signature:

Name: ViewRecommendation.

Input: Nothing.

Output: Table containing names of users.

\*/

CREATE PROCEDURE ViewRecommendation

AS

BEGIN

SELECT f\_Name , l\_name

FROM Users U

WHERE U.id NOT IN (SELECT user\_id FROM Recommendation);

END;

GO;

/\* 2.16 Malak

Create new note.

Signature:

Name: CreateNote.

Input: @User\_id int, @note\_id int, @title varchar(50), @Content varchar(500), @creation\_date

datetime.

Output: Nothing.

\*/

CREATE PROCEDURE CreateNote

@User\_id int,

@note\_id int,

@title varchar(50),

@Content varchar(500),

@date\_of\_creation datetime

AS

BEGIN

if @User\_id IS NOT NULL AND @note\_id IS NOT NULL

AND (EXISTS(SELECT \* FROM Users WHERE Users.id = @User\_id))

INSERT INTO Notes VALUES (@note\_id, @User\_id, @Content, @date\_of\_creation, @title)

ELSE

PRINT 'Please enter a valid user id and note id'

END;

go;

/\* 2.17 Malak

Receive a transaction.

Signature:

Name: ReceiveMoney.

Input: @sender\_id int, @type varchar(30), @amount decimal(13,2), @status varchar(10), @date

datetime.

Output: Nothing.

\*/

CREATE PROCEDURE ReceiveMoney

@receiver\_id int,

@type varchar(30),

@amount decimal(13,2),

@status varchar(10),

@date datetime

AS

BEGIN

IF @receiver\_id IS NOT NULL

AND (EXISTS(SELECT \* FROM Users WHERE Users.id = @receiver\_id))

INSERT INTO Finance(user\_id, type, amount ,status, date)

VALUES (@receiver\_id, @type, @amount, @status, @date)

ELSE

PRINT 'Please enter a valid user id'

-- insert one record

END;

GO

/\* 2.18 Malak

Create a payment on a specific date.

Signature:

Name: PlanPayment.

Input: @sender\_id int, @reciever\_id int , @amount decimal(13,2), @status varchar(10), @deadline datetime.

Output: Nothing

\*/

CREATE PROCEDURE PlanPayment

@sender\_id int,

@reciever\_id int ,

@amount decimal(13,2),

@status varchar(10),

@deadline datetime

AS

BEGIN

if @sender\_id IS NOT NULL AND @reciever\_id IS NOT NULL

AND (EXISTS(SELECT \* FROM Users WHERE Users.id = @sender\_id))

AND (EXISTS(SELECT \* FROM Users WHERE Users.id = @reciever\_id))

INSERT INTO Finance(user\_id, type, amount ,status, deadline)

VALUES (@sender\_id, 'outgoing', @amount, @status, @deadline),

(@reciever\_id, 'ingoing', @amount, @status , @deadline)

ELSE

PRINT 'Please enter a valid sender id and receiver id'

-- insert two record

END;

GO

GO

/\* 2.19 Phebe

Send message to user.

Signature:

Name: SendMessage.

Input: @sender\_id int, @receiver\_id int, @title varchar(30), @content Varchar(200), @timesent

time, @timereceived time.

Output: Nothing.

\*/

CREATE PROCEDURE SendMessage

@sender\_id int,

@receiver\_id int,

@title varchar(30),

@content Varchar(200),

@timesent datetime,

@timereceived datetime

AS

BEGIN

IF EXISTS (select \* from Users where id=@sender\_id ) AND EXISTS (select \* from Users where id=@receiver\_id )

INSERT INTO Communication (sender\_id, reciever\_id ,content, time\_sent , time\_received , title)

VALUES( @sender\_id, @receiver\_id ,@content,@timesent,@timereceived,@title)

ELSE

print 'sender or receiver invalid'

END;

GO

/\* 2.20 Malak

Change note title for all notes user created.

Signature:

Name: NoteTitle.

Input: @user\_id int,@note\_title varchar(50).

Output: Nothing.

\*/

CREATE PROCEDURE NoteTitle

@user\_id int,@note\_title varchar(50)

AS

BEGIN

IF @user\_id IS NOT NULL AND @note\_title IS NOT NULL

AND (EXISTS(SELECT \* FROM Users WHERE Users.id = @user\_id))

UPDATE Notes

SET title = @note\_title

WHERE user\_id = @user\_id

ELSE

PRINT 'Please enter a valid user id and note title'

END;

GO

/\* 2.21 Phebe

Show all messages received from a spacific user.

Signature:

Name: ShowMessages.

Input: @user\_id int, @sender\_id int.

Output: Table Containing all details of these massages.

\*/

CREATE PROCEDURE ShowMessages

@user\_id int,

@sender\_id int

AS

BEGIN

SELECT \*

FROM Communication

WHERE (reciever\_id = @user\_id AND sender\_id = @sender\_id);

END;

GO

/\* 3.1 Toqa

See details of all users and filter them by @user\_type

Signature:

Name: ViewUsers.

Input: @user\_type varchar(20)

Output: table containing details of the users

\*/

CREATE PROCEDURE ViewUsers

@user\_type varchar(20)

AS

BEGIN

IF @user\_type = 'Guest'

SELECT \*

FROM Users

WHERE type='Guest';

else

SELECT \*

FROM Users

WHERE type='Admin'

END;

GO;

/\* 3.2 Sarah

Remove an event from the system.

Signature:

Name: RemoveEvent.

Input: @event\_id int, @user\_id int.

Output: Nothing.

\*/

CREATE PROCEDURE RemoveEvent

@event\_id int,

@user\_id int

AS

BEGIN

IF NOT EXISTS (SELECT 1 FROM Calendar WHERE event\_id = @event\_id AND user\_assigned\_to=@user\_id)

PRINT 'event does not exist'

ELSE

DELETE

FROM Calendar

WHERE event\_id = @event\_id AND user\_assigned\_to=@user\_id;

END;

GO;

/\* 3.3 Phebe

Create schedule for the rooms.

Signature:

Name: CreateSchedule.

Input: @creator\_id int, @room\_id int, @start\_time datetime, @end\_time datetime, @action varchar(20).

Output: Nothing.

\*/

CREATE PROCEDURE CreateSchedule

@creator\_id int,

@room\_id int,

@start\_time datetime,

@end\_time datetime,

@action varchar(20)

AS

BEGIN

IF EXISTS (select \* from Admin where admin\_id=@creator\_id)

AND EXISTS (select \* from Room where room\_id=@room\_id)

INSERT INTO RoomSchedule(creator\_id, action, room, start\_time, end\_time)

VALUES (@creator\_id, @action, @room\_id, @start\_time, @end\_time);

ELSE

print 'user or room does not exist'

END;

GO;

/\* 3.4 Malak

Remove a guest from the system.

Signature:

Name: GuestRemove.

Input: @guest\_id int, @admin\_id int.

Output: @number\_of\_allowed \_guests int.

\*/

CREATE PROCEDURE GuestRemove

@guest\_id int,

@admin\_id int,

@number\_of\_allowed\_guests int OUTPUT

AS

BEGIN

IF @guest\_id IS NOT NULL AND @admin\_id IS NOT NULL

AND (EXISTS(SELECT \* FROM Guest WHERE Guest.guest\_id = @guest\_id))

AND (EXISTS(SELECT \* FROM Admin WHERE Admin.admin\_id = @admin\_id))

BEGIN

DELETE FROM Users

WHERE id = @guest\_id;

DECLARE @cnt INT;

SELECT @cnt = COUNT(\*)

From Guest

WHERE guest\_of = @admin\_id;

DECLARE @limit INT;

SELECT @limit = no\_of\_guests\_allowed

FROM Admin

WHERE admin\_ID = @admin\_id;

SET @number\_of\_allowed\_guests = @limit - @cnt;

END

ELSE

PRINT 'Please enter a valid guest id and admin id'

END;

GO

/\* 3.5 Malak

Recommend travel destinations for guests under certain age.

Signature:

Name: RecommendTD.

Input: @Guest\_id int, @destination varchar(10), @age int , @preference\_no int.

Output: Nothing

\*/

CREATE PROCEDURE RecommendTD

@Guest\_id int,

@destination varchar(10),

@age int,

@preference\_no int

AS

BEGIN

IF @Guest\_id IS NOT NULL

AND @preference\_no IS NOT NULL

AND EXISTS(SELECT \* FROM Guest WHERE Guest.guest\_id = @Guest\_id)

AND EXISTS(SELECT \* FROM Preferences WHERE Preferences.preference\_no = @preference\_no AND user\_id=@guest\_id)

BEGIN

IF EXISTS(SELECT \* FROM Users WHERE Users.id = @Guest\_id AND Users.age < @age)

INSERT INTO Recommendation VALUES (@Guest\_id,'Travel', @preference\_no, @destination)

ELSE

PRINT 'Please enter a valid age'

END

ELSE

PRINT 'Please enter a valid guest id and preference number'

--SELECT destination

--FROM Travel INNER JOIN User\_trip ON Travel.trip\_no = User\_trip.trip\_no INNER JOIN [User].ID ON

--check on age

END;

GO

/\* 3.6 Toqa

Access cameras in the house.

Signature:

Name: Servailance.

Input: @user\_id int, @location int,@camera\_id int .

Output: Nothing.

\*/

CREATE PROCEDURE Servailance

@user\_id int, @location int,@camera\_id int

AS

BEGIN

IF EXISTS (select \* from Camera where monitor\_id=@user\_id)

print 'Monitor already exists'

ELSE

BEGIN

IF EXISTS (select \* from Admin where admin\_id=@user\_id) AND EXISTS (select \* from Room where room\_id=@location)

AND EXISTS (select \* from Camera where camera\_id=@camera\_id)

INSERT INTO Camera(monitor\_id,camera\_id,room\_id)

VALUES (@user\_id,@camera\_id,@location)

ELSE

print 'user or location or camera dont exist'

END;

END;

GO;

/\* 3.7 Phebe

Change status of room.

Signature:

Name: RoomAvailability

Input: @location int, @status varchar(40)

Output: Nothing.

\*/

CREATE PROCEDURE RoomAvailability

@location int,

@status varchar(40)

AS

BEGIN

UPDATE Room

SET status = @status

WHERE room\_id =@location

END;

GO;

/\* 3.8 Toqa

Create an inventory for a specific item.

Signature:

Name:Sp\_ Inventory.

Input: @item\_id int„@name varchar(30), @quantity int, @expirydate datetime, @price decimal(10,2),

@manufacturer varchar(30),@category varchar(20)

Output: Nothing.

\*/

CREATE PROCEDURE Sp\_Inventory

@item\_id int,

@name varchar(30),

@quantity int,

@expirydate datetime,

@price decimal(10,2),

@manufacturer varchar(30),

@category varchar(20)

AS

BEGIN

INSERT INTO Inventory(supply\_id , name ,quantity,expiry\_date ,price , manufacturer ,category )

VALUES (@item\_id ,@name , @quantity , @expirydate , @price , @manufacturer ,@category)

END;

GO;

/\* 3.9 Toqa

Calculate price of purchasing a certain item.

Signature:

Name: Shopping.

Input: @id int , @quantity int , @total\_price decimal(10,2).

Output: @total\_price decimal(10,2).

\*/

CREATE PROCEDURE Shopping

@id int ,

@quantity int ,

@total\_price decimal(10,2) OUTPUT

AS

BEGIN

DECLARE @new\_price decimal(10,2)

SELECT @new\_price = price

FROM Inventory

WHERE supply\_id = @id

SET @total\_price = @new\_price \* @quantity

END;

GO;

/\* 3.10 Phebe

If current user had an activity set its duration to 1 hour.

Signature:

Name: LogActivityDuration .

Input: @room\_id int ,@device\_id int, @user\_id int,@date datetime, @duration varchar(50).

Output: Nothing.

\*/

CREATE PROCEDURE LogActivityDuration

@room\_id int ,

@device\_id int,

@user\_id int,

@date datetime,

@duration varchar(50)

AS

BEGIN

UPDATE Log

SET duration = @duration

WHERE room\_id = @room\_id AND

device\_id =@device\_id AND

user\_id = @user\_id AND

date = @date AND

activity IS NOT NULL

END;

GO;

/\* 3.11 Toqa

Set device consumption for all tablets.

Signature:

Name: TabletConsumption.

Input: @consumption int.

Output: Nothing.

\*/

CREATE PROCEDURE TabletConsumption

@consumption int

AS

BEGIN

UPDATE C

SET C.consumption = @consumption

FROM Consumption C

INNER JOIN Device D ON C.device\_id = D.device\_id

WHERE D.type = 'Tablet';

END;

GO;

/\* 3.12 Phebe

Make preferences for Room temperature to be 30 if a user is older then 40 .hint : ussenestedquery

Signature:

Name: MakePreferencesRoomTemp.

Input: @user\_id int ,@category varchar(20), @preferences\_number int .

Output: Nothing.

\*/

CREATE PROCEDURE MakePreferencesRoomTemp

@user\_id int ,

@category varchar(20),

@preferences\_number int

AS

BEGIN

IF EXISTS(SELECT \* FROM Users WHERE id =@user\_id AND age >40)

INSERT INTO Preferences (user\_id, preference\_no,category,content)

VALUES (@user\_id,@preferences\_number,@category, 'Room temperature: 30');

ELSE

print 'normal temperature no change'

END;

GO;

/\* 3.13 Phebe

View Log entries involving the user.

Signature:

Name: ViewMyLogEntry.

Input: @user\_id int.

Output: Table containing all the details of the log.

\*/

CREATE PROCEDURE ViewMyLogEntry

@user\_id int

AS

BEGIN

SELECT \*

FROM Log

WHERE user\_id = @user\_id

END;

GO;

/\* 3.14 Phebe

Update log entries involving the user.

Signature:

Name: UpdateLogEntry.

Input: @user\_id int, @room\_id int, @device\_id int,@activity varchar(30).

Output: Nothing.

\*/

CREATE PROCEDURE UpdateLogEntry

@user\_id int,

@room\_id int,

@device\_id int,

@activity varchar(30)

AS

BEGIN

IF EXISTS (select \* from Room where room\_id=@room\_id) AND EXISTS (select \* from Device where device\_id=@device\_id)

UPDATE Log

SET activity = @activity,

room\_id = @room\_id,

device\_id = @device\_id

WHERE user\_id = @user\_id;

ELSE

print 'room or device do not exist'

END;

GO;

/\* 3.15 Phebe

View rooms that are not being used.

Signature:

Name: ViewRoom.

Input: Nothing.

Output: Table containing all details of the Rooms.

\*/

CREATE PROCEDURE ViewRoom

AS

BEGIN

SELECT \*

FROM Room

WHERE status <> 'Booked'

END;

Go;

/\* 3.16 Phebe

View the details of the booked rooms given @user\_id and @room\_id . (If @room\_id is not booked

then show all rooms that are booked by this user).

Signature:

Name: ViewMeeting.

Input: @room\_id int, @user\_id int.

Output: Table containing all details of the activity and table containing name of each use

\*/

CREATE PROCEDURE ViewMeeting

@room\_id int,

@user\_id int

AS

BEGIN

IF EXISTS (SELECT \* FROM LOG WHERE room\_id = @room\_id AND user\_id = @user\_id)

BEGIN

SELECT U.f\_Name, U.l\_Name, L.activity, L.date, L.duration

FROM Log L

INNER JOIN Users U ON L.user\_id = U.id

WHERE L.room\_id = @room\_id AND L.user\_id = @user\_id;

END

ELSE

BEGIN

SELECT room\_id

FROM Log L

WHERE user\_id = @user\_id;

END

END;

GO

/\* 3.17 Toqa

Add to the tasks.

Signature:

Name: AdminAddTask.

Input: @user\_id int , @creator int ,@name varchar(30), @category varchar(20), @priority Int,@status

varchar(20), @reminder datetime , @deadline datetime , @other\_user varchar(20), @deadline datetime.

Output: Nothing.

\*/

CREATE PROCEDURE AdminAddTask

@user\_id int ,

@creator int ,

@name varchar(30),

@category varchar(20),

@priority Int,

@status varchar(20),

@reminder datetime ,

@deadline datetime ,

@other\_user int

---there is 2 deadline , so we removed one of them

AS

BEGIN

IF EXISTS (select \* from Admin where admin\_id = @creator)

BEGIN

INSERT INTO Task(name ,creation\_date,due\_date,category,creator ,status,reminder\_date,priority)

VALUES (@name,CURRENT\_TIMESTAMP,@deadline,@category,@creator ,@status,@reminder,@priority)

declare @task\_id int

select top 1 @task\_id=task\_id from Task where creator = @creator order by task\_id desc

print @task\_id

IF EXISTS (select 1 from Users where id = @user\_id)

INSERT INTO Assigned\_to VALUES (@creator, @task\_id, @user\_id)

ELSE

print 'user does not exist'

IF @other\_user IS NOT NULL AND EXISTS (select \* from Users where id=@other\_user)

INSERT INTO Assigned\_to VALUES (@creator, @task\_id, @other\_user)

ELSE

print 'other user does not exist'

END

ELSE

print 'creator does not exist'

END;

GO;

/\* 3.18 Sarah

Add Guests to the system , generate passwords for them and reserve rooms under their name.

Signature:

Name: AddGuest.

Input: @email varchar(30), @first\_name varchar(10) ,@address varchar (30),@password varchar(30),@guest\_of

int,@room\_id int

Output: @number\_of\_allowed \_guests in

\*/

CREATE PROCEDURE AddGuest

@email varchar(50),

@first\_name varchar(20),

@last\_name varchar(20),

@address varchar(30),

@password varchar(10),

@guest\_of int,

@room\_id int,

@birth\_date datetime,

@number\_of\_allowed\_guests int OUTPUT

AS

BEGIN

DECLARE @admin\_exists int;

SELECT @admin\_exists = COUNT(\*)

FROM Users

WHERE id = @guest\_of AND type = 'Admin';

IF @admin\_exists = 0

BEGIN

PRINT 'Admin does not exist';

END

ELSE

BEGIN

SELECT @number\_of\_allowed\_guests = COUNT(\*)

FROM Guest

WHERE guest\_of = @guest\_of;

IF (@number\_of\_allowed\_guests < (SELECT no\_of\_guests\_allowed FROM Admin WHERE admin\_id = @guest\_of))

BEGIN

INSERT INTO Users (f\_Name, l\_Name, password, email, room, type, birthdate)

VALUES (@first\_name, @last\_name, @password, @email, @room\_id, 'Guest', @birth\_date);

DECLARE @user\_id int;

SELECT @user\_id = id

FROM Users

WHERE f\_Name = @first\_name AND l\_Name = @last\_name AND email = @email AND password = @password AND room = @room\_id AND birthdate = @birth\_date;

INSERT INTO Guest (guest\_id, guest\_of, address)

VALUES (@user\_id, @guest\_of, @address);

SELECT @number\_of\_allowed\_guests = COUNT(\*)

FROM Guest

WHERE guest\_of = @guest\_of;

SET @number\_of\_allowed\_guests = (SELECT no\_of\_guests\_allowed FROM Admin WHERE admin\_id = @guest\_of) - @number\_of\_allowed\_guests;

END

ELSE

BEGIN

PRINT 'No more guests allowed';

  END

  END

END;

GO;

/\* 3.19 Sarah

Assign task to a specific User.

Signature:

Name: AssignTask.

Input: @user\_id int , @task\_id int , @creator\_id int.

Output: Nothing.

\*/

CREATE PROCEDURE AssignTask

@user\_id int,

@task\_id int,

@creator\_id int

AS

BEGIN

IF EXISTS (SELECT 1 FROM Task WHERE Task\_id = @task\_id AND creator=@creator\_id)

AND EXISTS (SELECT 1 FROM Users WHERE Users.id=@user\_id)

INSERT INTO Assigned\_to VALUES (@creator\_id, @task\_id, @user\_id)

ELSE

print 'task does not exist'

END;

GO;

/\* 3.20 Phebe

Delete last message sent.

Signature:

Name: DeleteMsg.

Input: Nothing.

Output: Nothing.

\*/

CREATE PROCEDURE DeleteMsg

AS

BEGIN

DELETE

FROM Communication

WHERE time\_sent = (SELECT MAX(time\_sent) FROM Communication);

END;

GO;

/\* 3.21 Malak

Add outgoing flight itinerary for a specific flight.

Signature:

Name: AddItinerary.

Input: @@trip\_no int,@flight\_num varchar(30) ,@flight\_date datetime ,@destination varchar(40).

Output: Nothing

\*/

CREATE PROCEDURE AddItinerary

@trip\_id int,

@flight\_num varchar(30) ,

@flight\_date datetime ,

@destination varchar(40)

AS

BEGIN

IF @trip\_id IS NOT NULL

UPDATE Travel

SET outgoing\_flight\_date = @flight\_date,

outgoing\_flight\_num = @flight\_num,

destination = @destination

WHERE trip\_no = @trip\_id

ELSE

PRINT 'Please enter a valid trip id'

END;

GO

/\* 3.22 Malak

Change flight date to next year for all flights in current year.

Signature:

Name: ChangeFlight.

Input: Nothing.

Output: Nothing

\*/

CREATE PROCEDURE ChangeFlight

AS

BEGIN

UPDATE Travel

SET ingoing\_flight\_date = DATEADD(YEAR, 1, ingoing\_flight\_date)

WHERE YEAR(ingoing\_flight\_date) = YEAR(GETDATE());

UPDATE Travel

SET outgoing\_flight\_date = DATEADD(YEAR, 1, outgoing\_flight\_date)

WHERE YEAR(outgoing\_flight\_date) = YEAR(GETDATE());

END;

GO

/\* 3.23 Malak

Update incoming flights.

Signature:

Name: UpdateFlight.

Input: @date datetime, @time time, @destination varchar(15)

Output: Nothing

\*/

CREATE PROCEDURE UpdateFlight

@trip\_id INT,

@date datetime,

@destination varchar(15)

AS

BEGIN

IF @trip\_id IS NOT NULL

AND EXISTS(SELECT \* FROM Travel WHERE Travel.trip\_no = @trip\_id)

UPDATE Travel

SET ingoing\_flight\_date = @date, destination = @destination

WHERE trip\_no = @trip\_id

ELSE

PRINT 'Please enter a valid trip id'

END;

GO

/\* 3.24 Phebe

Add a new device.

Signature:

Name: AddDevice.

Input:@device\_id int, @status varchar(20), @battery int,@location int, @type varchar(20).

Output: Nothing.

\*/

CREATE PROCEDURE AddDevice

@device\_id int,

@status varchar(20),

@battery int,

@location int,

@type varchar(20)

AS

BEGIN

IF EXISTS (SELECT \* FROM Room where room\_id=@location)

INSERT INTO Device (device\_id, room, type, status, battery\_status)

VALUES (@device\_id,@location,@type, @status , @battery)

ELSE

print 'room does not exist'

END;

GO;

/\* 3.25 Malak

Find the location of all devices out of battery.

Signature:

Name: OutOfBattery.

Input: Nothing.

Output: List of rooms

\*/

CREATE PROCEDURE OutOfBattery

AS

BEGIN

SELECT Room.\*

FROM Device INNER JOIN Room ON Device.room = Room.room\_id

WHERE battery\_status = 0

END;

GO

/\* 3.26 Malak

Set the status of all devices out of battery to charging.

Signature:

Name: Charging.

Input: Nothing.

Output: Nothing

\*/

CREATE PROCEDURE Charging

AS

BEGIN

UPDATE Device

SET battery\_status =100

WHERE battery\_status = 0

END;

GO

/\* 3.27 Toqa

Set the number of allowed guests for an admin.

Signature:

Name: GuestsAllowed.

Input: @admin\_id int,@number\_of\_guests int

Output: Nothing.

\*/

CREATE PROCEDURE GuestsAllowed

@admin\_id int,

@number\_of\_guests int

AS

BEGIN

IF EXISTS (select 1 from Admin where admin\_id=@admin\_id )

UPDATE Admin

SET no\_of\_guests\_allowed = @number\_of\_guests

WHERE admin\_id = @admin\_id;

ELSE

print 'admin does not exist'

END;

GO;

/\* 3.28 Malak

Add a penalty for all unpaid transactions where the deadline has passed.

Signature:

Name: Penalize.

Input: @Penalty\_amount int.

Output: Nothing.

\*/

CREATE PROCEDURE Penalize

@Penalty\_amount int

AS

BEGIN

UPDATE Finance

SET penalty = @Penalty\_amount

WHERE deadline<GETDATE()

END;

GO

/\* 3.29 Toqa

Get the number of all guests currently present for a certain admin.

Signature:

Name: GuestNumber.

Input: @admin\_id int.

Output: The Number of all guests currently present .

\*/

CREATE PROCEDURE GuestNumber

@admin\_id int

AS

BEGIN

SELECT COUNT(\*)

FROM Admin A

INNER JOIN Guest G ON A.admin\_id = G.guest\_of

WHERE A.admin\_id = @admin\_id;

END;

GO

/\* 3.30 Toqa

Get the youngest user in the system (hint : use limit).

Signature:

Name: Youngest.

Input: Nothing.

Output: The youngest user in the system.

\*/

CREATE PROCEDURE Youngest

AS

BEGIN

SELECT TOP 1 f\_Name ,l\_Name ,age

FROM Users

ORDER BY age ASC

END;

GO;

/\* 3.31 Malak

Get the users whose average income per month is greater then a specific amount.

Signature:

Name: AveragaPayment.

Input: @amount decimal (10,2).

Output: Table Containing users names

\*/

CREATE PROCEDURE AveragePayment

@amount decimal (10,2)

AS

BEGIN

SELECT Users.f\_name,Users.l\_name

FROM Users

INNER JOIN Admin ON Users.id = Admin.admin\_id

WHERE Admin.salary > @amount;

END;

go

/\* 3.32 Toqa

Get sum the sum of all purchases needed in the home inventory (assuming you need only 1 of each

missing item) .

Signature:

Name: Purchase.

Input: Nothing.

Output: The sum amount

\*/

CREATE PROCEDURE Purchase

AS

BEGIN

SELECT SUM(price)

FROM Inventory

WHERE quantity < 1;

END;

GO;

/\* 3.33 Phebe

Get the location where more then two devices have a dead battery.

Signature:

Name: NeedCharge.

Input: Nothing.

Output: Table Containing locations.

\*/

CREATE PROCEDURE NeedCharge

AS

BEGIN

--NOTE: WE CANNOT SELECT DEVICE ID ALONE BECAUSE OF THE GROUP BY,

--WE CAN ONLY SELECT THE GROUPED BY ATTRIBUTE AND AN AGGREGATE FUNCTION

SELECT room

FROM Device

WHERE battery\_status = 0

GROUP BY room

HAVING COUNT(device\_id) >2;

END;

GO;

/\* 3.34 Sarah

Get the admin with more then 2 guests.

Signature:

Name: Admins.

Input: Nothing.

Output: Table containing Admins names.

\*/

CREATE PROCEDURE Admins

AS

BEGIN

SELECT u.f\_name, u.l\_name

FROM Users u

INNER JOIN (

SELECT a.admin\_id

FROM Admin a

INNER JOIN Guest g ON a.admin\_id = g.guest\_of

GROUP BY a.admin\_id

HAVING COUNT(g.guest\_of) > 2

) AS a ON u.id = a.admin\_id;

END;

GO

INSERTIONS

use HomeSync

INSERT INTO Room

VALUES

('living', 1, 'free'),

('bedroom', 2, 'occupied'),

('bedroom', 3, 'occupied'),

('bedroom', 2, 'occupied'),

('bedroom', 3, 'occupied'),

('bedroom', 4, 'free'),

('kitchen', 1, 'free'),

('garden', 0, 'free');

INSERT INTO Users

VALUES

('Sarah', 'El-Feel', '12345678', 'sarah.elfeel@gmail.com', 'dark mode', 1, 'Admin', '05/10/2004'),

('Malak', 'Abdelbaki', 'abcdefgh', 'malak.ismail@gmail.com', 'no preference', 8, 'Admin', '09/28/2004'),

('Toqa', 'Amr', 'hello123', 'toqa.amr@gmail.com', 'no preference', 1, 'Admin', '11/06/2004'),

('Phebe', 'Atef', 'database10', 'phebe.atef@gmail.com', 'default preference', 3, 'Admin', '08/10/2004'),

('Ahmed', 'Mohamed', 'project@40', 'ahmed123@gmail.com', 'light mode', 4, 'Admin', '11/11/2002'),

('Kareem', 'Ahmed', 'Kareem2005', 'kareem.ahmed@gmail.com', 'default', 1, 'Guest', '11/04/2005'),

('Mariam', 'Ahmed', 'Mariam2010', 'mariam.ahmed@gmail.com', 'no preference', 5, 'Guest', '06/04/2010'),

('Mariam', 'Eid', 'Eid2004', 'mariam.eid@gmail.com', 'no preference', 6, 'Guest', '12/04/2004'),

('Eyad', 'Hesham', 'NoRiskNoFu', 'eyad@gmail.com', 'default', 7, 'Guest', '09/21/2004'),

('Zeina', 'Ismail', 'Zonzon9', 'zeina@gmail.com', 'dark mode', 4, 'Guest', '09/07/2004'),

('Mustafa', 'Ahmed', 'GIU123', 'mustafa@hotmail.com', 'default', 2, 'Guest', '09/07/1970');

INSERT INTO Admin

VALUES

(1, 4, 1000),

(2, 2, 2000),

(3, 1, 2500.5),

(4, 3, 1920),

(5, 1, 1891);

INSERT INTO Guest

VALUES

(6, 1, 'Cairo', '12/12/2023', '05/05/2024', 0),

(7, 1, 'Giza', '11/11/2023', '06/05/2025', 1),

(8, 2, 'Augsburg', '04/04/2023', '09/09/2027', 1),

(9, 1, 'Mannheim', '06/06/2023', '08/08/2027', 1),

(10, 4, 'Cairo', '07/07/2023', '01/01/2024', 1),

(11,5, 'Dubai', '1/1/2023', '12/12/2024', 0);

INSERT INTO Task

VALUES

('Math', '11/18/2023', '12/04/2023', 'Assignment', 1, 'not done', '12/03/2023', 1),

('Database M3', '01/01/2024', '01/31/2024', 'Project', 2, 'not done', '01/29/2024', 2),

('Prog', '11/01/2023', '11/15/2023', 'Project', 3, 'not done', '11/14/2023', 3);

INSERT INTO Assigned\_to

VALUES

(2, 2, 8);

INSERT INTO Calendar

VALUES

(1, 9, 'Christmas', 'Festival', 'Berlin', '12/23/2023'),

(2, 1, 'Finals', 'Exams', 'Cairo', '01/01/2024');

INSERT INTO Notes

VALUES

(1, 1, 'flight tickets', '12/27/2023', 'Travel'),

(2, 6, 'studying', '09/09/2023', 'Uni');

INSERT INTO Travel

VALUES

(1, 'Kempinski', 'Dubai', 182, 188, '02/03/2024', '02/09/2024', 'Dubai', 'Cairo', 'Bus'),

(2, 'Fairmont', 'Cairo', 363, 918, '05/05/2022', '06/06/2022', 'Cairo', 'Berlin', 'Uber'),

(3, 'Novotel', 'Munich', '102', '103', '1/1/2023', '12/12/2023', 'Cairo', 'Munich', 'Bus');

INSERT INTO User\_trip

VALUES (1, 1, 2, 128,818 )

INSERT INTO RoomSchedule

VALUES

(1, 'Karaoke', 1, '12/04/2023', '12/05/2023'),

(8, 'Cooking', 6, '11/04/2023', '11/05/2023');

INSERT INTO Device

VALUES

(1, 2, 'Desktop', 'free', 90),

(2,3, 'Laptop', 'occupied', 81),

(3,4, 'Smart TV', 'free', 72),

(4,3, 'Ipad', 'occupied', 0),

(5,1, 'Smart TV', 'free', 81),

(7,3, 'tv', 'in use', 0),

(8,3,'ipad', 'in use', 0),

(9, 1, 'Tablet', 'free', 100);

INSERT INTO Log

VALUES

(3, 2, 4, 'studying', '11/24/2023', 5);

INSERT INTO Communication

VALUES

(1, 6, 'HBD', '11/04/2023', '11/04/2023', '11/04/2023', 'birthday'),

(4, 2, 'Which page', '05/05/2022', '05/06/2022', '05/06/2022', 'work');

INSERT INTO Finance

VALUES

(1, 'outgoing', 1872, 'EGP', 'deposit', 'accepted', '07/07/2023', 161615, '07/31/2023', 0),

(7, 'ingoing', 167511, 'USD', 'transfer', 'accepted', '08/31/2022', 52542, '09/20/2023', 0);

INSERT INTO Health

VALUES

(1, '11/05/2023', 'cycling', 5, 'pasta'),

(2, '11/06/2023', 'gym', 8, ' bread');

INSERT INTO Preferences VALUES (6,1,'studying', 'study at night');

INSERT INTO Recommendation VALUES (6,'UNI',1,'start at 8');

INSERT INTO Inventory

VALUES

(1, 'oil', 2, '11/11/2024', 30, 'Crystal', 'Food'),

(2, 'light bulbs', 0, '12/12/2024', 77, 'Philips', 'Electronics'),

(3, 'chipsy', 0, '12/4/2023', 10, 'Lion', 'Food');

INSERT INTO Camera

VALUES

(1, 1, 6),

(5, 2, 4);

INSERT INTO Consumption

VALUES

('11/11/2023', 3, 120),

('4/4/2023', 2, 570),

('12/12/2024', 9, 9999);

EXECUTIONS

use HomeSync

--SECTION 1: UNREGISTERED USER--------

--1.1

DECLARE @user\_id INT;

EXEC UserRegister 'Guest', 'moahmed@gmail.com', 'Mohamed ', 'Ibrahim', '9/28/1975', 'QRSTU', @user\_id OUTPUT;

PRINT @user\_id;

--SECTION 2: REGISTERED USER----------

--2.1

DECLARE @user\_id INT;

DECLARE @success BIT;

EXEC UserLogin 'moahmed@gmail.com', 'QRSTU', @success OUTPUT, @user\_id OUTPUT;

PRINT @user\_id;

PRINT @success;

--2.2

EXEC ViewProfile 6

--2.4

EXEC ViewMyTask 3;

--2.5

EXEC FinishMyTask 1, 'Math';

--2.6

EXEC ViewTask 8, 2;

--2.7

declare @charge int

declare @location int

EXEC ViewMyDeviceCharge 2, @charge output, @location output

print @charge

print @location

--2.8

EXEC AssignRoom 9, 8

--2.9

EXEC CreateEvent 3, 2, 'Eid', 'festival', 'Cairo', '06/06/2024', 6

--2.10

EXEC AssignUser 7, 2

--2.11

EXEC AddReminder 1, '12/12/2024'

--2.12

EXEC Uninvited 2, 7

--2.13

EXEC UpdateTaskDeadline '01/02/2024', 3

--2.14

EXEC ViewEvent 9, 1

--2.15

EXEC ViewRecommendation

--2.16

EXEC CreateNote 6, 3, 'HOMEWORK' , 'CPS', '1/1/2020'

--2.17

EXEC ReceiveMoney 6, 'deposit', 1000,'incoming', '1/1/2020'

--2.18

EXEC PlanPayment 6,8, 1000,'accepted', '1/1/2020'

--2.19

declare @date datetime

set @date = CURRENT\_TIMESTAMP

exec SendMessage 3, 4, 'Hi hello', 'message', @date, '11/25/2023'

--2.20

exec NoteTitle 6, 'assignment'

--2.21

exec ShowMessages 6, 1

--SECTION 3: REGISTERED ADMIN ------------------

--3.1

EXEC ViewUsers 'Admin'

--3.2

EXEC RemoveEvent 3, 2

--3.3

exec CreateSchedule 2, 8, '11/24/2023', '11/25/2023', 'studying'

--3.4

declare @cnt int;

exec GuestRemove 11, 5, @cnt output;

print @cnt;

--3.5

EXEC RecommendTD 6, 'Bahamas', 19, 1

--3.6

EXEC Servailance 2, 7, 1

--3.7

exec RoomAvailability 8, 'free'

--3.8

EXEC Sp\_Inventory 4, 'milk', 1, '12/12/2023', 20, 'Juhayna' , 'Food'

--3.9

declare @total\_price DECIMAL(10,2)

EXEC Shopping 2, 2, @total\_Price OUTPUT

PRINT @total\_price

--3.10

exec LogActivityDuration 3, 2, 4, '11/24/2023', '20'

--3.11

EXEC TabletConsumption 50

--3.12

exec MakePreferencesRoomTemp 11, 'Temperature', 2

--3.13

exec ViewMyLogEntry 4

--3.14

exec UpdateLogEntry 4, 3, 2, 'sleeping'

--3.15

exec ViewRoom

--3.16

exec ViewMeeting 3, 4

--3.17

EXEC AdminAddTask 6, 2, 'play', 'basketball', 1, 'not done', '12/24/2023', '12/25/2023', 9

--3.18

DECLARE @nrGuests int

EXEC AddGuest 'youssef@gmail.com', 'Youssef', 'Ahmed', 'Sheikh Zayed', 'Yoyo2015',2, 7, '12/13/2000', @nrGuests OUTPUT

PRINT @nrGuests

--3.19

EXEC AssignTask 9, 2, 2

--3.20

exec DeleteMsg

--3.21

EXEC AddItinerary 1, 200, '10/2/2024','Rome'

--3.22

EXEC ChangeFlight

--3.23

EXEC UpdateFlight 1, '4/4/2024','London'

--3.24

exec AddDevice 6, 'in use', 99, 8, 'SmartPhone'

--3.25

EXEC OutOfBattery

--3.26

EXEC Charging

--3.27

EXEC GuestsAllowed 5, 4

--3.28

EXEC Penalize 1000

--3.29

EXEC GuestNumber 1

--3.30

EXEC Youngest

--3.31

EXEC AveragePayment 1900

--3.32

EXEC Purchase

--3.33

exec NeedCharge

--3.34

exec Admins