American International University – Bangladesh

Spring 2022-23 [Mid-Term]

Advance Database Management System

Section – B

Course Teacher:

Juena Ahmed Noshin

Project Name:

Spouse Finding Management System

Group Members:

Sl No	Name	Student ID	Contribution	
01	MD. ASIFUR RAHMAN	20-41898-1	Class, Use Case and Activity Diagram	20%
02	SABIHA KHAIR OHI	20-41905-1	User Interface Design, Schema Diagram and Sequence	20%
03	TANVIR ANZOOM	20-41920-1	Project Proposal, Query Writing, Relational Algebra, Conclusion	20%
04	RIJOAN FARDOUS	20-41943-1	Scenario Description, ER Diagram and Normalization	20%
05	FABIHA TASNIM TRISHA	20-42829-1	Introduction, Table Creation and Data Insertion	20%

Table of Contents

Introduction

A spouse finding management system is a software application that helps users to find their potential life partners based on various criteria, such as age, location, education, occupation, interests, and more. This system can be used by anyone who is looking for a spouse, whether they are single or divorced. The main purpose of this system is to provide an easy and efficient way for users to search for and connect with potential partners.

The spouse finding management system typically consists of a user-friendly interface that allows users to create a profile, search for potential partners, and communicate with them through messaging or chat features. The system may also offer additional features such as advanced search options, personality matching, and privacy settings.

In the project report, you can introduce the spouse finding management system by discussing the need for such a system in today's society, where traditional methods of finding a spouse may not be as effective or efficient. You can also highlight the benefits of using a software application for spouse finding, such as the ability to search for partners based on specific criteria and to connect with potential partners from different regions or cultures.

Additionally, you can discuss the features and functionality of your spouse finding management system, such as the user interface, search capabilities, matching algorithms, and communication tools. You can also mention any unique or innovative features of your system that sets it apart from other spouse finding platforms.

Project Proposal

Project Summary:

In today's world, finding a compatible life partner is becoming increasingly challenging. With the advent of technology and the internet, online dating platforms have become very popular. However, these platforms often lack a comprehensive system to match individuals based on their personality traits, values, and beliefs. This is where the Spouse Finding Management System comes in.

The Spouse Finding Management System is a web-based platform that allows users to create a profile and find potential partners based on their preferences. The system uses a matching algorithm that takes into account a user's personality traits, values, and beliefs to suggest compatible partners. Users can communicate with potential partners within the platform and arrange to meet in person if they feel comfortable.

Objectives:

The primary objectives of the Spouse Finding Management System are as follows:

- Provide a platform for users to find compatible life partners based on their personality traits, values, and beliefs.
- Develop a matching algorithm that accurately matches users with potential partners.
- Create a secure and user-friendly platform that ensures the privacy and safety of users.
- Facilitate communication between users to help them get to know each other better.
- Develop a revenue stream through subscription-based memberships.

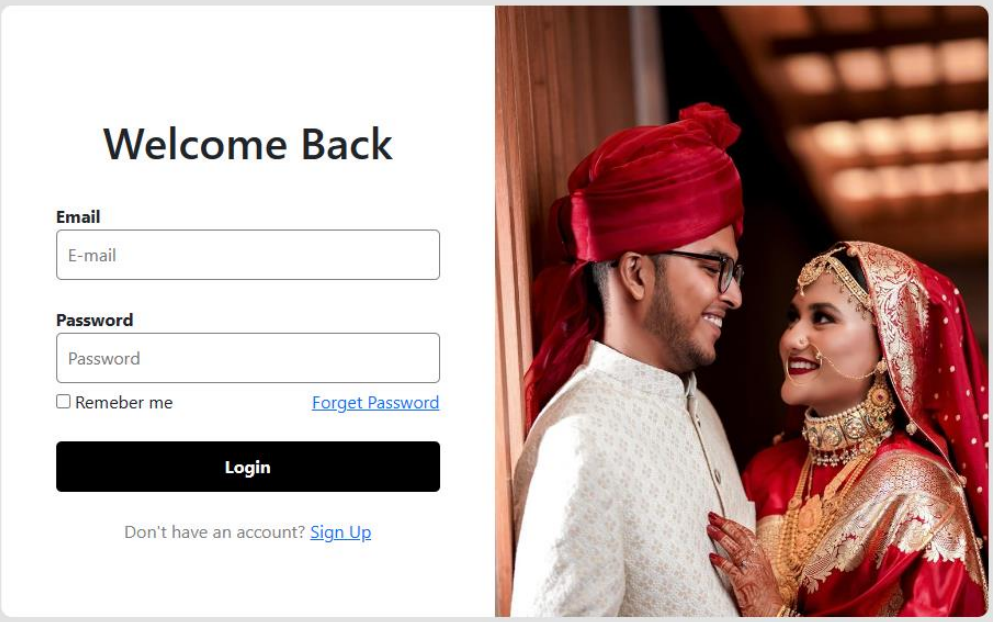
Features:

The Spouse Finding Management System will include the following features:

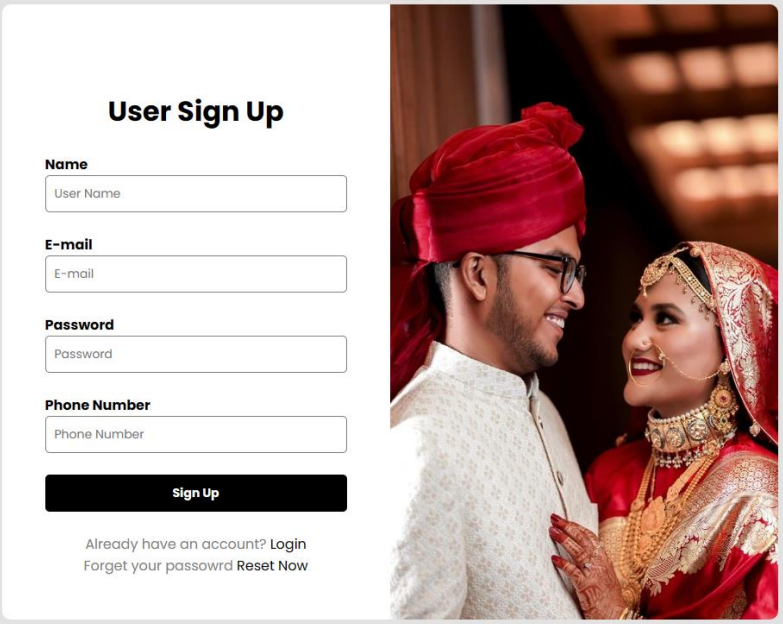
- User profiles: Users can create a profile that includes their personal information, such as age, occupation, education, and interests.
- Matching algorithm: The system will use a matching algorithm that takes into account a user's personality traits, values, and beliefs to suggest compatible partners.
- Messaging system: Users can communicate with potential partners within the platform.
- Video calls: Users can have video calls with potential partners to get to know them better.
- Search filters: Users can search for potential partners based on various criteria, such as age, location, occupation, and interests.

User Interface

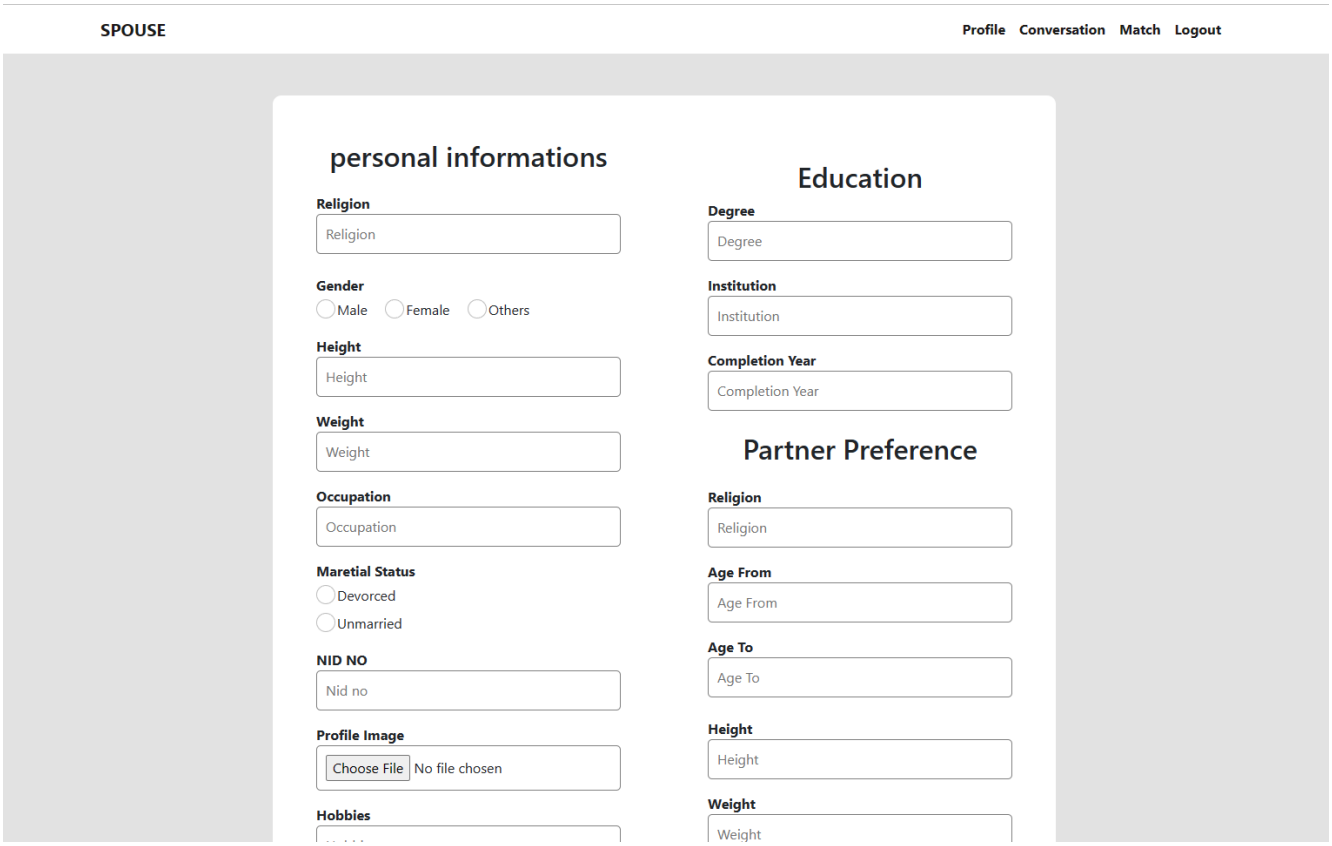
Login Page:



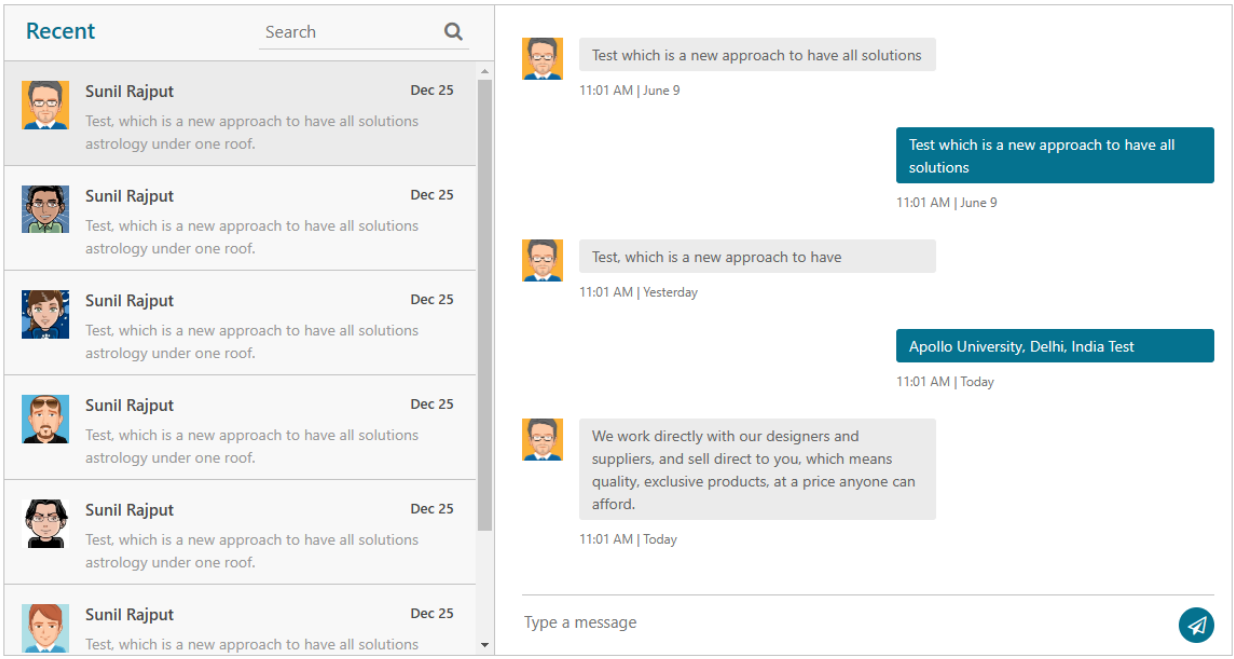
Signup Page:



Profile Page:

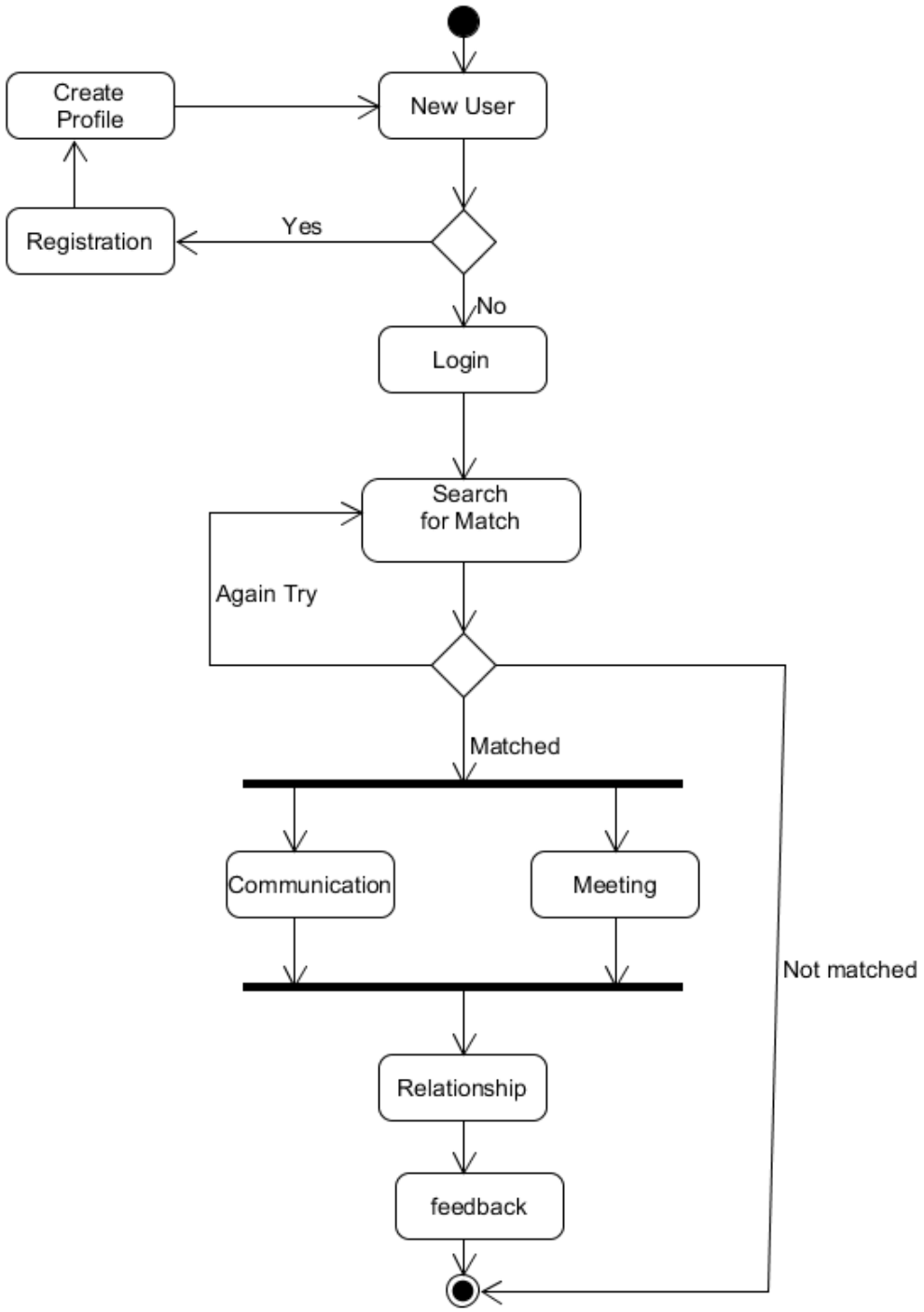


Conversation Page:

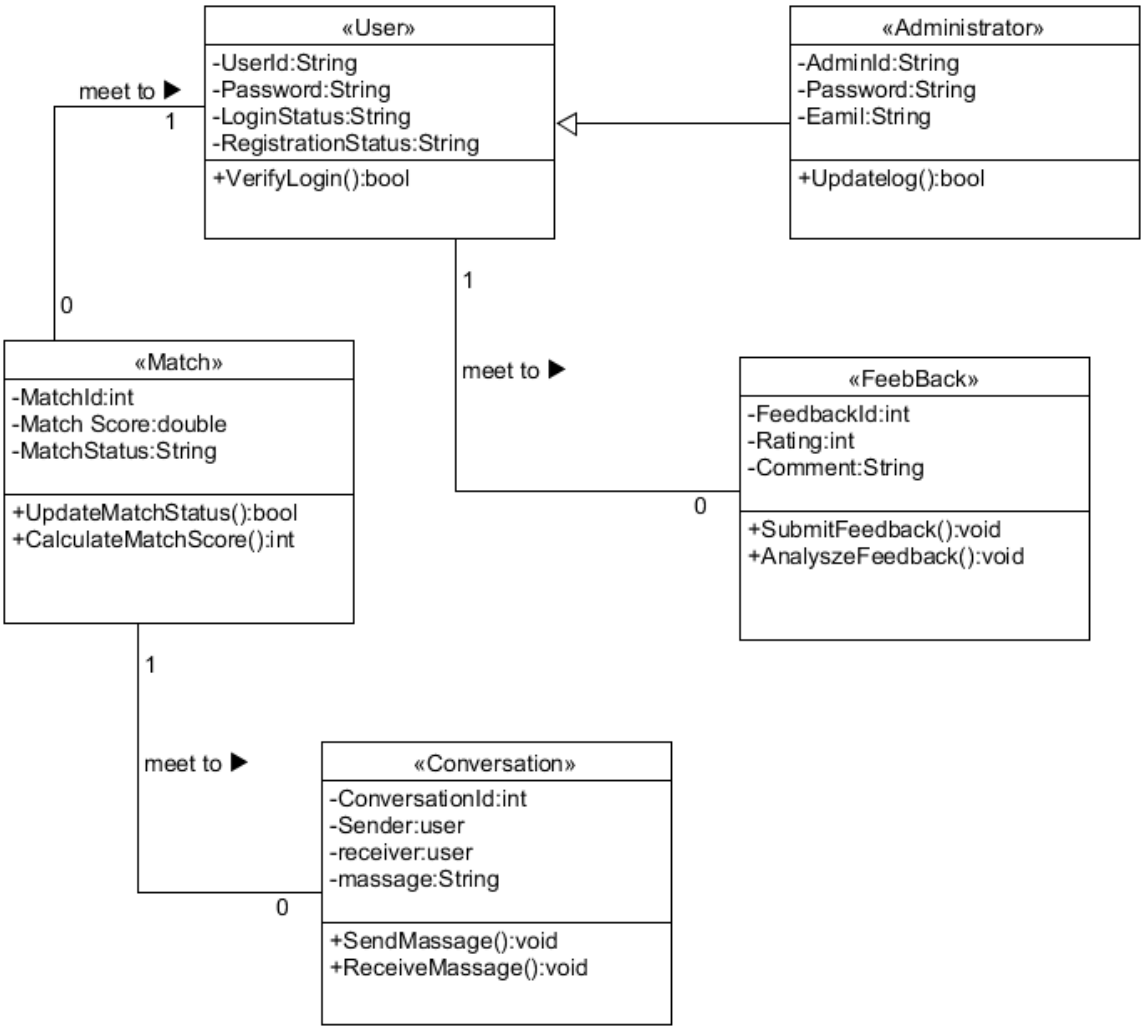


Diagrams

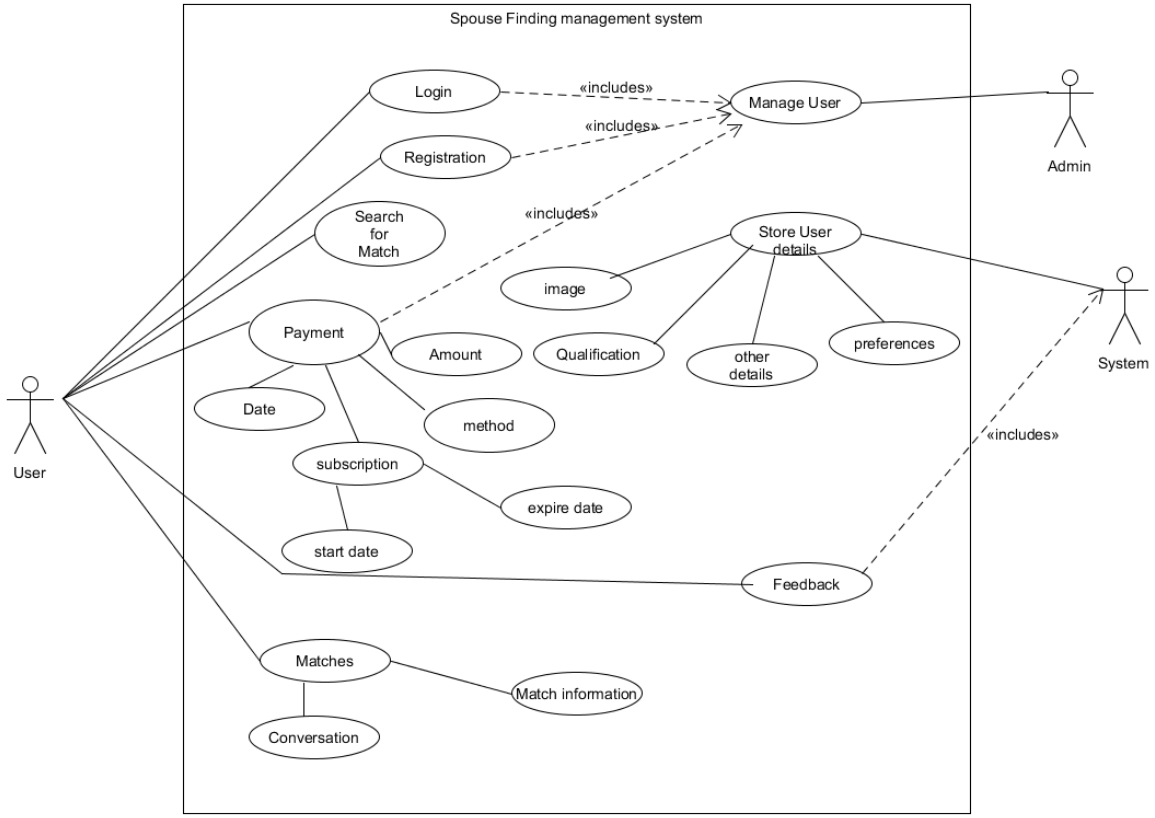
Activity Diagram:



Class Diagram:



Use Case Diagram:



Scenario Description

In a Spouse Finding management system, a user can create a profile. A profile is only available to a single user. Each user has a unique user id. User data such as name, date of birth, age, gender, email, password, phone, nil and address are also stored in the system. Age is calculated from the date of birth. A user address is composed of house number, street number, zip code and city. A unique profile id identifies a profile. The system also stores images, religion, occupation, educational qualification, height, weight, a brief description, hobbies, marital status and preferences. A profile may have multiple images and hobbies. The Educational Qualification consists of the degree, institution, and year of completion. Religion, occupation, height, weight, hobbies, age range, location, and marital status make up the user's preferences.

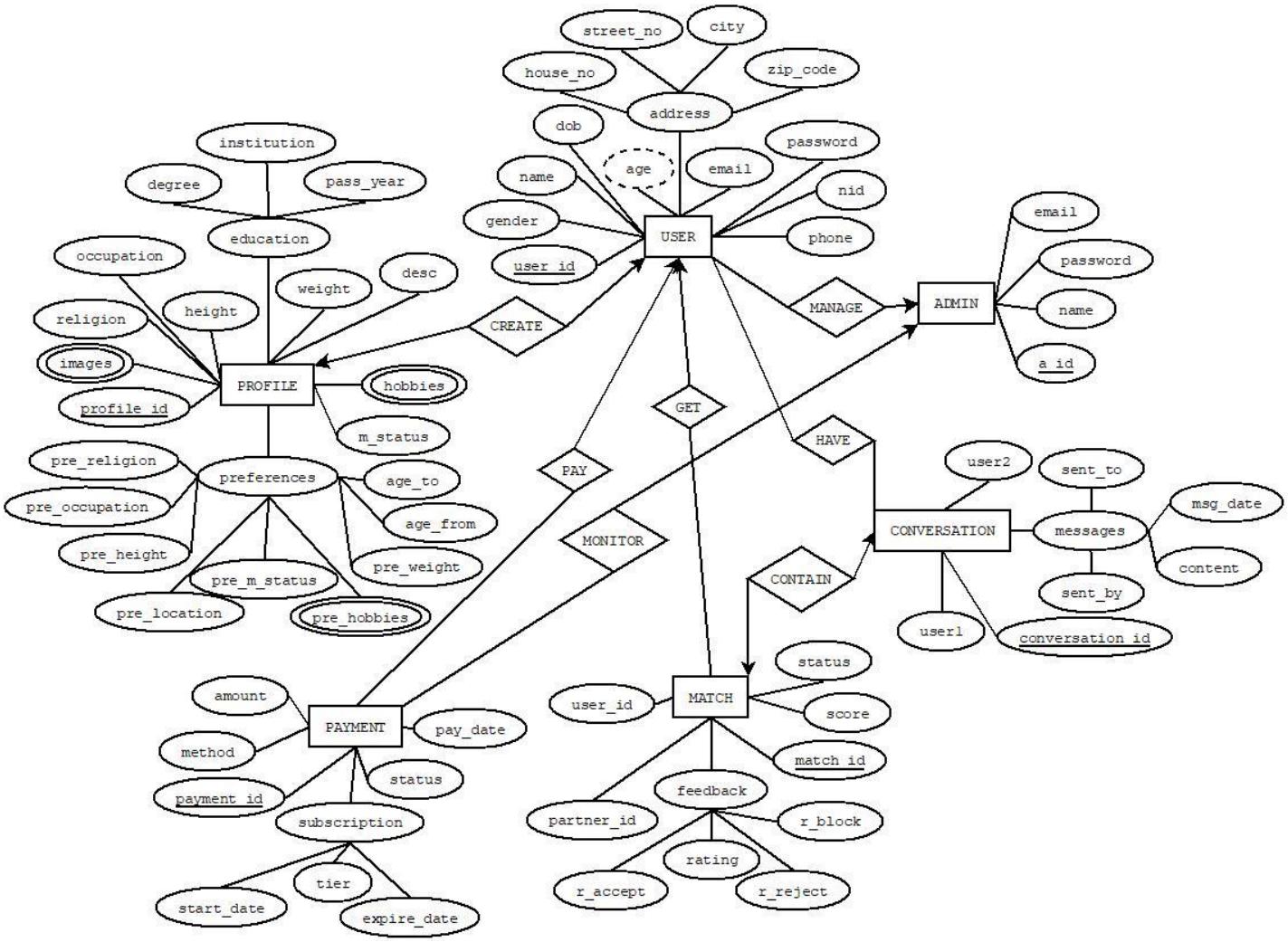
An admin can manage many users. But one user is managed by one admin. A unique admin id identifies an admin. Admin data such as name, email, and password is also stored in the system.

To use the service, one user may make multiple payments but each payment is made by only one user. Each payment has a unique payment id. Payment method, payment amount, payment date, payment status, subscription plan are also stored in the system. A subscription plan consists of tier, start date, expire date. An admin can monitor many payments but each payment is monitored by an admin.

A user can have many matches. But each match belongs to only one user. A unique match id identifies a match. Other information such as user id, partner id, match score, match date, match status and feedback is also stored in the system. A feedback is composed of opinion, reason of acceptance, reason of rejection, reason of block and match ratings.

Each match has a conversation and each conversation belongs to one match. A conversation id identifies each conversation. Id of both users and messages are also stored in the system. Each message is composed of content (text/image), sent by, sent to and date. A user can have many conversations and each conversation has two users.

ER Diagram


street_no
city
house_no
zip_code
address
dob
password
age
email
name
nid
gender
USER
phone
user_id
email
password
MANAGE
ADMIN
name
a_id
institution
degree
pass_year
education
occupation
desc
weight
height
religion
CREATE
images
PROFILE
hobbies
profile_id
m_status
GET
HAVE
pre_religion
preferences
age_to
PAY
user2
sent_to
pre_occupation
age_from
MONITOR
CONVERSATION
messages
msg_date
pre_height
pre_weight
pre_m_status
CONTAIN
content
sent_by
pre_location
pre_hobbies
user1
conversation_id
status
amount
user_id
MATCH
score
PAYMENT
pay_date
method
match_id
status
payment_id
feedback
subscription
partner_id
r_block
rating
tier
r_accept
r_reject
start_date
expire_date

Normalization

Normalization Steps:

CREATE BRANCH (User - Profile)

UNF:

create(user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, profile_id, image, religion, occupation, degree, institution, pass_year, height, weight, desc, hobbies, marital_status, pre_religion, pre_occupation, pre_height, pre_weight, pre_hobbies, age_from, age_to, pre_location, pre_marital_status)

1NF:

images, hobbies and pre_hobbies are multivalued attributes.

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, profile_id, image, religion, occupation, degree, institution, pass_year, height, weight, desc, hobbies, marital_status, pre_religion, pre_occupation, pre_height, pre_weight, pre_hobbies, age_from, age_to, pre_location, pre_marital_status

2NF:

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city
2. profile_id, image, religion, occupation, degree, institution, pass_year, height, weight, desc, hobbies, marital_status, pre_religion, pre_occupation, pre_height, pre_weight, pre_hobbies, age_from, age_to, pre_location, pre_marital_status

3NF:

1. user_id, name, dob, age, gender, email, password, phone, nid
2. house_no, street_no, zip_code, city
3. profile_id, image, religion, occupation, height, weight, desc, hobbies, marital_status
4. degree, institution, pass_year
5. pre_religion, pre_occupation, pre_height, pre_weight, pre_hobbies, age_from, age_to, pre_location, pre_marital_status

Table Creation:

1. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
2. add_id, house_no, street_no, zip_code, city
3. profile_id, image, religion, occupation, height, weight, desc, hobbies, marital_status, **edu_id, user_id**
4. edu_id, degree, institution, pass_year
5. pre_id, pre_religion, pre_occupation, pre_height, pre_weight, pre_hobbies, age_from, age_to, pre_location, pre_marital_status, **profile_id**

MANAGE BRANCH (Admin - User)

UNF:

manage(user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, admin_id, name, email, password)

1NF:

There is no multivalued attribute. The Relation is already 1NF.

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, admin_id, name, email, password

2NF:

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city
2. admin_id, name, email, password

3NF:

1. user_id, name, dob, age, gender, email, password, phone, nid,
2. house_no, street_no, zip_code, city
3. admin_id, name, email, password

Table Creation:

1. user_id, name, dob, age, gender, email, password, phone, nid, **add_id, admin_id**
2. add_id, house_no, street_no, zip_code, city
3. admin_id, name, email, password

PAY BRANCH (User - Payment)

UNF:

pay(user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, payment_id, tier, start_date, expire_date, method, amount, pay_date, status)

1NF:

There is no multivalued attribute. Relation is already 1NF.

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, payment_id, tier, start_date, expire_date, method, amount, pay_date, status

2NF:

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city
2. payment_id, tier, start_date, expire_date, method, amount, pay_date, status

3NF:

1. user_id, name, dob, age, gender, email, password, phone, nid
2. house_no, street_no, zip_code, city
3. payment_id, method, amount, pay_date, status
4. tier, start_date, expire_date

Table Creation:

1. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
2. add_id, house_no, street_no, zip_code, city
3. payment_id, method, amount, pay_date, status, **user_id, sub_id**
4. sub_id, tier, start_date, expire_date

MONITOR BRANCH (Admin - Payment)

UNF:

monitor(admin_id, name, email, password, payment_id, tier, start_date, expire_date, method, amount, pay_date, status)

1NF:

There is no multivalued attribute. Relation is already 1NF.

1. admin_id, name, email, password, payment_id, tier, start_date, expire_date, method, amount, pay_date, status

2NF:

1. admin_id, name, email, password
2. payment_id, tier, start_date, expire_date, method, amount, pay_date, status

3NF:

1. admin_id, name, email, password
2. payment_id, method, amount, pay_date, status
3. tier, start_date, expire_date

Table Creation:

1. admin_id, name, email, password
2. payment_id, method, amount, pay_date, status, **admin_id, sub_id**
3. sub_id, tier, start_date, expire_date

GET BRANCH (User - Match)

UNF:

get(user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, match_id, user_id, partner_id, score, status, rating, r_accept, r_reject, r_block)

1NF:

There is no multivalued attribute. Relation is already 1NF.

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, match_id, user_id, partner_id, score, status, rating, r_accept, r_reject, r_block

2NF:

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city,
2. match_id, user_id, partner_id, score, status, rating, r_accept, r_reject, r_block

3NF:

1. user_id, name, dob, age, gender, email, password, phone, nid
2. house_no, street_no, zip_code, city,
3. match_id, user_id, partner_id, score, status
4. rating, r_accept, r_reject, r_block

Table Creation:

1. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
2. add_id, house_no, street_no, zip_code, city
3. match_id, user_id, partner_id, score, status, **feedback_id**
4. feedback_id, rating, r_accept, r_reject, r_block

CONTAIN BRANCH (Match - Conversation)

UNF:

contain(match_id, user_id, partner_id, score, status, rating, r_accept, r_reject, r_block, converstion_id, user1, user2, sent_by, sent_to, content, msg_date)

1NF:

There is no multivalued attribute. Relation is already 1NF.

1. match_id, user_id, partner_id, score, status, rating, r_accept, r_reject, r_block, converstion_id, user1, user2, sent_by, sent_to, content, msg_date

2NF:

1. match_id, user_id, partner_id, score, status, rating, r_accept, r_reject, r_block,
2. converstion_id, user1, user2, sent_by, sent_to, content, msg_date

3NF:

1. match_id, user_id, partner_id, score, status
2. rating, r_accept, r_reject, r_block,
3. converstion_id, user1, user2
4. sent_by, sent_to, content, msg_date

Table Creation:

1. match_id, user_id, partner_id, score, status, **feedback_id, conversation_id**
2. feedback_id, rating, r_accept, r_reject, r_block
3. converstion_id, user1, user2
4. msg_id, sent_by, sent_to, content, msg_date, **converstion_id**

HAVE BRANCH (User - Conversation)

UNF:

have(user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, converstion_id, user1, user2, sent_by, sent_to, content, msg_date)

1NF:

There is no multivalued attribute. Relation is already 1NF.

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city, converstion_id, user1, user2, sent_by, sent_to, content, msg_date

2NF:

1. user_id, name, dob, age, gender, email, password, phone, nid, house_no, street_no, zip_code, city,
2. converstion_id, user1, user2, sent_by, sent_to, content, msg_date

3NF:

1. user_id, name, dob, age, gender, email, password, phone, nid
2. house_no, street_no, zip_code, city,
3. converstion_id, user1, user2
4. sent_by, sent_to, content, msg_date

Table Creation:

1. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
2. add_id, house_no, street_no, zip_code, city
3. converstion_id, user1, user2
4. msg_id, sent_by, sent_to, content, msg_date, **converstion_id**

Temporary Tables:

1. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
2. add_id, house_no, street_no, zip_code, city
3. profile_id, image, religion, occupation, height, weight, desc, hobbies, marital_status, **edu_id, user_id**
4. edu_id, degree, institution, pass_year
5. pre_id, pre_religion, pre_occupation, pre_height, pre_weight, pre_hobbies, age_from, age_to, pre_location, pre_marital_status, **profile_id**
6. user_id, name, dob, age, gender, email, password, phone, nid, **add_id, admin_id**
7. add_id, house_no, street_no, zip_code, city
8. admin_id, name, email, password
9. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
10. add_id, house_no, street_no, zip_code, city
11. payment_id, method, amount, pay_date, status, **user_id, sub_id, admin_id**
12. sub_id, tier, start_date, expire_date
13. admin_id, name, email, password
14. payment_id, method, amount, pay_date, status, **admin_id, sub_id**
15. sub_id, tier, start_date, expire_date
16. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
17. add_id, house_no, street_no, zip_code, city
18. match_id, user_id, partner_id, score, status, **feedback_id**
19. feedback_id, rating, r_accept, r_reject, r_block
20. match_id, user_id, partner_id, score, status, **feedback_id, conversation_id**
21. feedback_id, rating, r_accept, r_reject, r_block
22. converstion_id, user1, user2
23. msg_id, sent_by, sent_to, content, msg_date, **converstion_id**
24. user_id, name, dob, age, gender, email, password, phone, nid, **add_id**
25. add_id, house_no, street_no, zip_code, city
26. converstion_id, user1, user2
27. msg_id, sent_by, sent_to, content, msg_date, **converstion_id**

Final Tables:

1. user_id, name, dob, age, gender, email, password, phone, nid, **add_id, admin_id**
2. add_id, house_no, street_no, zip_code, city
3. profile_id, image, religion, occupation, height, weight, desc, hobbies, marital_status, **edu_id, user_id**
4. edu_id, degree, institution, pass_year
5. pre_id, pre_religion, pre_occupation, pre_height, pre_weight, pre_hobbies, age_from, age_to, pre_location, pre_marital_status, **profile_id**
6. admin_id, name, email, password
7. payment_id, method, amount, pay_date, status, **user_id, sub_id, admin_id**
8. sub_id, tier, start_date, expire_date
9. match_id, user_id, partner_id, score, status, **feedback_id, conversation_id**
10. feedback_id, rating, r_accept, r_reject, r_block
11. converstion_id, user1, user2
12. msg_id, sent_by, sent_to, content, msg_date, **converstion_id**

Schema Diagram

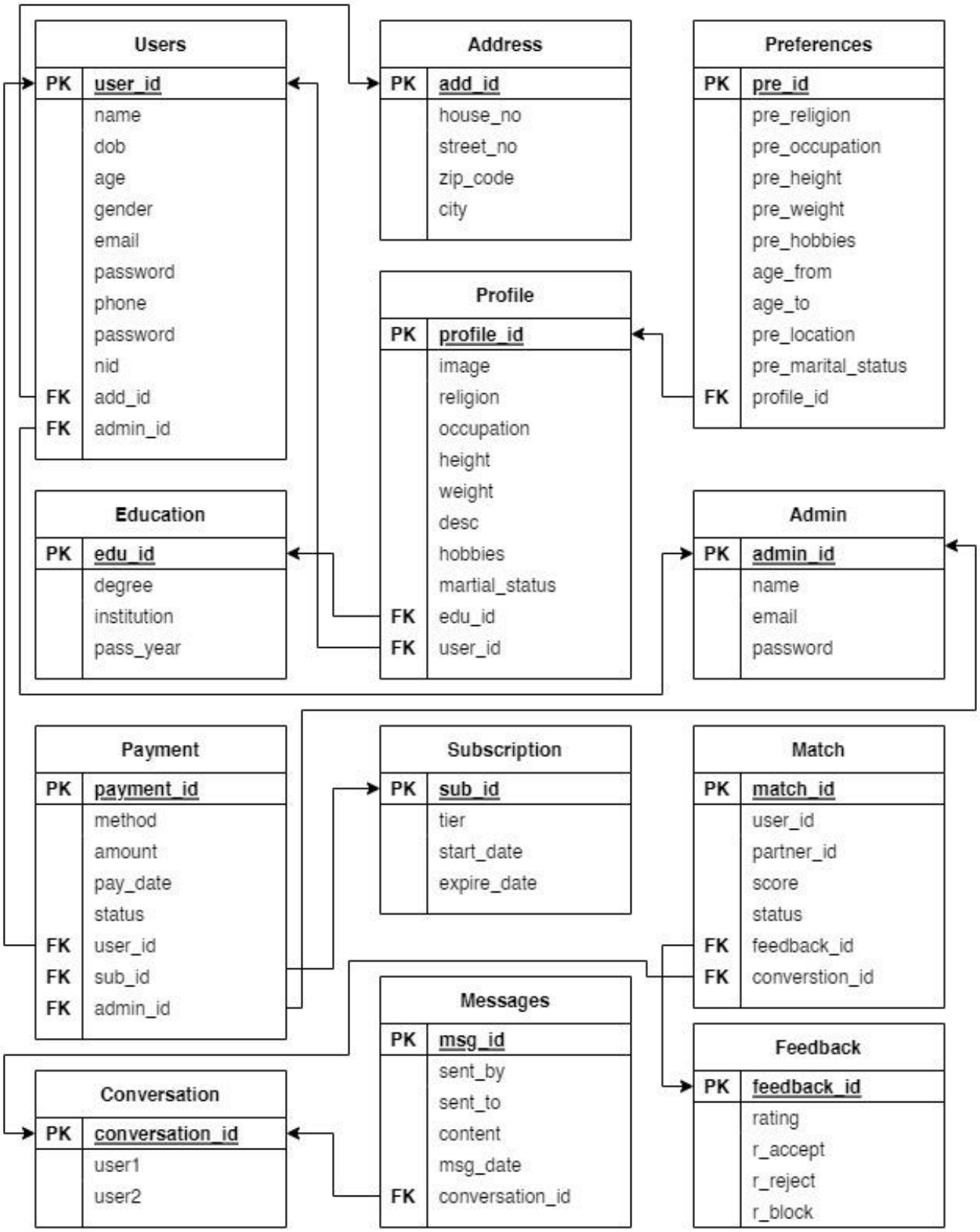


Table Creation

Address Table:

```
CREATE TABLE Address(add_id NUMBER (5) CONSTRAINT PK_Address PRIMARY KEY,
        house_no NUMBER (3),
        street_no VARCHAR2(20),
        zip_code NUMBER (4),
        city VARCHAR2 (15));
```

Object Type TABLE Object ADDRESS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ADDRESS	ADD_ID	Number	-	5	0	1	-	-	-
	HOUSE_NO	Number	-	3	0	-	✓	-	-
	STREET_NO	Varchar2	20	-	-	-	✓	-	-
	ZIP_CODE	Number	-	4	0	-	✓	-	-
	CITY	Varchar2	15	-	-	-	✓	-	-
									1 - 5

Admin Table:

```
CREATE TABLE Admin(admin_id NUMBER (5) CONSTRAINT PK_Admin PRIMARY KEY,
        name VARCHAR2(20),
        email VARCHAR2(20),
        password VARCHAR2 (20));
```

Object Type TABLE Object ADMIN

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ADMIN	ADMIN_ID	Number	-	5	0	1	-	-	-
	NAME	Varchar2	20	-	-	-	✓	-	-
	EMAIL	Varchar2	20	-	-	-	✓	-	-
	PASSWORD	Varchar2	20	-	-	-	✓	-	-
									1 - 4

Users Table:

```
CREATE TABLE Users(user_id NUMBER (5) CONSTRAINT PK_Users PRIMARY KEY,
        name VARCHAR2(20),
        dob DATE,
        age NUMBER (3),
        gender VARCHAR2(10),
```

phone NUMBER (11),
email VARCHAR2(20),
password VARCHAR2(20),
nid NUMBER (20),
add_id NUMBER (5) CONSTRAINT FK_add_id REFERENCES Address,
admin_id NUMBER (5) CONSTRAINT FK_admin_id REFERENCES Admin);

Object Type **TABLE** Object **USERS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
USERS	USER_ID	Number	-	5	0	1	-	-	-
	NAME	Varchar2	20	-	-	-	✓	-	-
	DOB	Date	7	-	-	-	✓	-	-
	AGE	Number	-	3	0	-	✓	-	-
	GENDER	Varchar2	10	-	-	-	✓	-	-
	PHONE	Number	-	11	0	-	✓	-	-
	EMAIL	Varchar2	20	-	-	-	✓	-	-
	PASSWORD	Varchar2	20	-	-	-	✓	-	-
	NID	Number	-	20	0	-	✓	-	-
	ADD_ID	Number	-	5	0	-	✓	-	-
	ADMIN_ID	Number	-	5	0	-	✓	-	-
									1 - 11

Education Table:

CREATE TABLE Education(edu_id NUMBER (5) CONSTRAINT PK_Education PRIMARY KEY,
degree VARCHAR2(20),
institution VARCHAR2(40),
pass_year NUMBER (4));

Object Type **TABLE** Object **EDUCATION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EDUCATION	EDU_ID	Number	-	5	0	1	-	-	-
	DEGREE	Varchar2	20	-	-	-	✓	-	-
	INSTITUTION	Varchar2	40	-	-	-	✓	-	-
	PASS_YEAR	Number	-	4	0	-	✓	-	-
									1 - 4

Profile Table:

CREATE TABLE Profile(profile_id NUMBER (5) CONSTRAINT PK_Profile PRIMARY KEY,

```
image VARCHAR2(30),
religion VARCHAR2(10),
occupation VARCHAR2(15),
height NUMBER(1,2),
weight NUMBER(3,3),
bio VARCHAR2(30),
hobby1 VARCHAR2(10),
hobby2 VARCHAR2(10),
marital_status VARCHAR2(10),
edu_id NUMBER (5) CONSTRAINT FK_edu_id REFERENCES Education,
user_id NUMBER (5) CONSTRAINT FK_user_id REFERENCES Users);
```

Object Type **TABLE** Object **PROFILE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PROFILE	PROFILE_ID	Number	-	5	0	1	-	-	-
	IMAGE	Varchar2	30	-	-	-	✓	-	-
	RELIGION	Varchar2	10	-	-	-	✓	-	-
	OCCUPATION	Varchar2	15	-	-	-	✓	-	-
	HEIGHT	Number	-	1	2	-	✓	-	-
	WEIGHT	Number	-	3	3	-	✓	-	-
	BIO	Varchar2	30	-	-	-	✓	-	-
	HOBBY1	Varchar2	10	-	-	-	✓	-	-
	HOBBY2	Varchar2	10	-	-	-	✓	-	-
	MARITAL_STATUS	Varchar2	10	-	-	-	✓	-	-
	EDU_ID	Number	-	5	0	-	✓	-	-
	USER_ID	Number	-	5	0	-	✓	-	-

1 - 12

Preferences Table:

```
CREATE TABLE Preferences(pre_id NUMBER (5) CONSTRAINT PK_Preferences
PRIMARY KEY,
pre_religion VARCHAR2(10),
pre_occupation VARCHAR2(15),
pre_height NUMBER(1,2),
pre_weight NUMBER(3,3),
age_from NUMBER(3),
age_to NUMBER(3),
pre_location VARCHAR2(10),
pre_hobby1 VARCHAR2(10),
pre_hobby2 VARCHAR2(10),
pre_marital_status VARCHAR2(10),
profile_id NUMBER (5) CONSTRAINT FK_profile_id REFERENCES Profile);
```

Object Type TABLE Object PREFERENCES

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PREFERENCES	PRE_ID	Number	-	5	0	1	-	-	-
	PRE_RELIGION	Varchar2	10	-	-	-	✓	-	-
	PRE_OCCUPATION	Varchar2	15	-	-	-	✓	-	-
	PRE_HEIGHT	Number	-	1	2	-	✓	-	-
	PRE_WEIGHT	Number	-	3	3	-	✓	-	-
	AGE_FROM	Number	-	3	0	-	✓	-	-
	AGE_TO	Number	-	3	0	-	✓	-	-
	PRE_LOCATION	Varchar2	10	-	-	-	✓	-	-
	PRE_HOBBY1	Varchar2	10	-	-	-	✓	-	-
	PRE_HOBBY2	Varchar2	10	-	-	-	✓	-	-
	PRE_MARITAL_STATUS	Varchar2	10	-	-	-	✓	-	-
	PROFILE_ID	Number	-	5	0	-	✓	-	-
									1 - 12

Subscription Table:

```
CREATE TABLE Subscription(sub_id NUMBER (5) CONSTRAINT PK_Subscription
PRIMARY KEY,
        tier VARCHAR2(10),
        start_date DATE,
        expire_date DATE);
```

Object Type TABLE Object SUBSCRIPTION

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SUBSCRIPTION	SUB_ID	Number	-	5	0	1	-	-	-
	TIER	Varchar2	10	-	-	-	✓	-	-
	START_DATE	Date	7	-	-	-	✓	-	-
	EXPIRE_DATE	Date	7	-	-	-	✓	-	-
									1 - 4

Payment Table:

```
CREATE TABLE Payment(payment_id NUMBER (5) CONSTRAINT PK_Payment PRIMARY
KEY,
        method VARCHAR2(10),
        amount NUMBER (5),
        pay_date DATE,
        status VARCHAR2(10),
```

user_id NUMBER (5) CONSTRAINT FK_user1_id REFERENCES Users,
sub_id NUMBER (5) CONSTRAINT FK_sub_id REFERENCES Subscription,
admin_id NUMBER (5) CONSTRAINT FK_admin1_id REFERENCES Admin);

Object Type **TABLE** Object **PAYMENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT	PAYMENT_ID	Number	-	5	0	1	-	-	-
	METHOD	Varchar2	10	-	-	-	✓	-	-
	AMOUNT	Number	-	5	0	-	✓	-	-
	PAY_DATE	Date	7	-	-	-	✓	-	-
	STATUS	Varchar2	10	-	-	-	✓	-	-
	USER_ID	Number	-	5	0	-	✓	-	-
	SUB_ID	Number	-	5	0	-	✓	-	-
	ADMIN_ID	Number	-	5	0	-	✓	-	-
									1 - 8

Conversation Table:

CREATE TABLE Conversation(conversation_id NUMBER (5) CONSTRAINT PK_Conversation PRIMARY KEY,
user1 NUMBER (5),
user2 NUMBER (5));

Object Type **TABLE** Object **CONVERSATION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CONVERSATION	CONVERSATION_ID	Number	-	5	0	1	-	-	-
	USER1	Number	-	5	0	-	✓	-	-
	USER2	Number	-	5	0	-	✓	-	-
									1 - 3

Messages Table:

CREATE TABLE Messages(msg_id NUMBER (5) CONSTRAINT PK_Messages PRIMARY KEY,
sent_by NUMBER (5),
sent_to NUMBER (5),
msg_date DATE,
content VARCHAR2(30),

conversation_id NUMBER (5) CONSTRAINT FK_conversation_id REFERENCES Conversation);

Object Type TABLE Object MESSAGES

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MESSAGES	MSG_ID	Number	-	5	0	1	-	-	-
	SENT_BY	Number	-	5	0	-	✓	-	-
	SENT_TO	Number	-	5	0	-	✓	-	-
	MSG_DATE	Date	7	-	-	-	✓	-	-
	CONTENT	Varchar2	30	-	-	-	✓	-	-
	CONVERSATION_ID	Number	-	5	0	-	✓	-	-
									1 - 6

Feedback Table:

CREATE TABLE Feedback(feedback_id NUMBER (5) CONSTRAINT PK_Feedback PRIMARY KEY,
rating NUMBER (1,1),
r_accept VARCHAR2(20),
r_block VARCHAR2(20),
r_reject VARCHAR2(20));

Object Type TABLE Object FEEDBACK

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FEEDBACK	FEEDBACK_ID	Number	-	5	0	1	-	-	-
	RATING	Number	-	1	1	-	✓	-	-
	R_ACCEPT	Varchar2	20	-	-	-	✓	-	-
	R_BLOCK	Varchar2	20	-	-	-	✓	-	-
	R_REJECT	Varchar2	20	-	-	-	✓	-	-
									1 - 5

Match Table:

```
CREATE TABLE Match(match_id NUMBER (5) CONSTRAINT PK_Match PRIMARY KEY,
        user_id NUMBER (5),
        partner_id NUMBER (5),
        score NUMBER (3),
        status VARCHAR2(10),
        feedback_id NUMBER (5) CONSTRAINT FK_feedback_id REFERENCES Feedback,
        conversation_id NUMBER (5) CONSTRAINT FK_conversation1_id REFERENCES

Conversation);
```

Object Type **TABLE** Object **MATCH**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MATCH	MATCH_ID	Number	-	5	0	1	-	-	-
	USER_ID	Number	-	5	0	-	✓	-	-
	PARTNER_ID	Number	-	5	0	-	✓	-	-
	SCORE	Number	-	3	0	-	✓	-	-
	STATUS	Varchar2	10	-	-	-	✓	-	-
	FEEDBACK_ID	Number	-	5	0	-	✓	-	-
	CONVERSATION_ID	Number	-	5	0	-	✓	-	-
									1 - 7

Sequences

Address Sequence:
```
CREATE SEQUENCE Address_add_id
        INCREMENT BY 2
        START WITH 10
        MAXVALUE 1000
        NOCACHE
        NOCYCLE;
```

Admin Sequence:
```
CREATE SEQUENCE Admin_admin_id
        INCREMENT BY 10
        START WITH 50
        MAXVALUE 5000
        NOCACHE
        NOCYCLE;
```

Users Sequence:
```
CREATE SEQUENCE Users_user_id
```

```
INCREMENT BY 1
START WITH 30
MAXVALUE 300
CACHE 10
NOCYCLE;
```

Education Sequence: CREATE SEQUENCE Education_edu_id

```
INCREMENT BY 3
START WITH 15
MAXVALUE 450
NOCACHE
NOCYCLE;
```

Profile Sequence: CREATE SEQUENCE Profile_pro_id

```
INCREMENT BY 4
START WITH 40
MAXVALUE 1200
NOCACHE
NOCYCLE;
```

Preferences Sequence: CREATE SEQUENCE Preferences_pre_id

```
INCREMENT BY 6
START WITH 90
MAXVALUE 900
NOCACHE
NOCYCLE;
```

Subscription Sequence: CREATE SEQUENCE Subscription_sub_id

```
INCREMENT BY 9
START WITH 2
MAXVALUE 3000
NOCACHE
NOCYCLE;
```

Payment Sequence: CREATE SEQUENCE Payment_pay_id

```
INCREMENT BY 7
START WITH 4
MAXVALUE 3000
NOCACHE
NOCYCLE;
```

Conversation Sequence: CREATE SEQUENCE Conversation_con_id

```
INCREMENT BY 8
START WITH 5
MAXVALUE 4000
NOCACHE
```

NOCYCLE;

Messages Sequence: CREATE SEQUENCE Messages_msg_id
INCREMENT BY 1
START WITH 110
MAXVALUE 9999
CACHE 20
NOCYCLE;

Feedback Sequence: CREATE SEQUENCE Feedback_fd_id
INCREMENT BY 2
START WITH 5
MAXVALUE 4000
NOCACHE
NOCYCLE;

Match Sequence: CREATE SEQUENCE Match_match_id
INCREMENT BY 9
START WITH 78
MAXVALUE 5000
NOCACHE
NOCYCLE;

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
ADDRESS_ADD_ID	1000	2	10
ADMIN_ADMIN_ID	5000	10	50
USERS_USER_ID	300	1	30
EDUCATION_EDU_ID	450	3	15
PROFILE_PRO_ID	1200	4	40
PREFERENCES_PRE_ID	900	6	90
SUBSCRIPTION_SUB_ID	3000	9	2
PAYMENT_PAY_ID	3000	7	4
CONVERSATION_CON_ID	4000	8	5
MESSAGES_MSG_ID	9999	1	110
FEEDBACK_FD_ID	4000	2	5
MATCH_MATCH_ID	5000	9	78

Data Insertion

Address Table:

1. INSERT INTO ADDRESS VALUES (Address_add_id.NEXTVAL, 12, 'ABC', 1215, 'DHAKA');
2. INSERT INTO ADDRESS VALUES (Address_add_id.NEXTVAL, 23, 'BCD', 1326, 'COMILLA');
3. INSERT INTO ADDRESS VALUES (Address_add_id.NEXTVAL, 34, 'CDE', 1437, 'KHULNA');
4. INSERT INTO ADDRESS VALUES (Address_add_id.NEXTVAL, 45, 'DEF', 1548, 'SYLHET');
5. INSERT INTO ADDRESS VALUES (Address_add_id.NEXTVAL, 56, 'EFG', 1659, 'DHAKA');

ADD_ID	HOUSE_NO	STREET_NO	ZIP_CODE	CITY
10	12	ABC	1215	DHAKA
12	23	BCD	1326	COMILLA
14	34	CDE	1437	KHULNA
16	45	DEF	1548	SYLHET
18	56	EFG	1659	DHAKA

Admin Table:

1. INSERT INTO ADMIN VALUES (Admin_admin_id.NEXTVAL, 'Asif', 'asif@sfms.com', 'asif');
2. INSERT INTO ADMIN VALUES (Admin_admin_id.NEXTVAL, 'Ohi', 'ohi@sfms.com', 'ohi');
3. INSERT INTO ADMIN VALUES (Admin_admin_id.NEXTVAL, 'Siam', 'siam@sfms.com', 'siam');
4. INSERT INTO ADMIN VALUES (Admin_admin_id.NEXTVAL, 'Rijoan', 'rijoan@sfms.com', 'rijoan');
5. INSERT INTO ADMIN VALUES (Admin_admin_id.NEXTVAL, 'Fabiha', 'fabiha@sfms.com', 'fabiha');

ADMIN_ID	NAME	EMAIL	PASSWORD
50	Asif	asif@sfms.com	asif
60	Ohi	ohi@sfms.com	ohi
70	Siam	siam@sfms.com	siam
80	Rijoan	rijoan@sfms.com	rijoan
90	Fabiha	fabiha@sfms.com	fabiha

Users Table:

1. INSERT INTO USERS VALUES (Users_user_id.NEXTVAL, 'User 1', to_date('01-01-2000','dd-mm-yyyy'), 23, 'Female', 01114841851, 'user1@gmail.com', 'user1', 65454484145, 10, 50);
2. INSERT INTO USERS VALUES (Users_user_id.NEXTVAL, 'User 2', to_date('02-02-2004','dd-mm-yyyy'), 19, 'Female', 01914484185, 'user2@gmail.com', 'user2', 543535434, 12, 60);
3. INSERT INTO USERS VALUES (Users_user_id.NEXTVAL, 'User 3', to_date('03-03-2003','dd-mm-yyyy'), 20, 'Male', 01294841884, 'user3@gmail.com', 'user3', 5348343453, 14, 50);
4. INSERT INTO USERS VALUES (Users_user_id.NEXTVAL, 'User 4', to_date('04-04-2002','dd-mm-yyyy'), 21, 'Female', 01114841485, 'user4@gmail.com', 'user4', 45383483, 16, 70);
5. INSERT INTO USERS VALUES (Users_user_id.NEXTVAL, 'User 5', to_date('05-05-2001','dd-mm-yyyy'), 22, 'Male', 01754841851, 'user5@gmail.com', 'user2', 39837838, 18, 90);

USER_ID	NAME	DOB	AGE	GENDER	PHONE	EMAIL	PASSWORD	NID	ADD_ID	ADMIN_ID
30	User 1	01-JAN-00	23	Female	1114841851	user1@gmail.com	user1	65454484145	10	50
32	User 2	02-FEB-04	19	Female	1914484185	user2@gmail.com	user2	543535434	12	60
33	User 3	03-MAR-03	20	Male	1294841884	user3@gmail.com	user3	5348343453	14	50
34	User 4	04-APR-02	21	Female	1114841485	user4@gmail.com	user4	45383483	16	70
35	User 5	05-MAY-01	22	Male	1754841851	user5@gmail.com	user2	39837838	18	90

5 rows returned in 0.00 seconds CSV Export

Education Table:

1. INSERT INTO EDUCATION VALUES (Education_edu_id.NEXTVAL, 'BSc CSE', 'AIUB', 2023);
2. INSERT INTO EDUCATION VALUES (Education_edu_id.NEXTVAL, 'BSc CSE', 'NSU', 2024);

3. INSERT INTO EDUCATION VALUES (Education_edu_id.NEXTVAL, 'BSc CSE', 'IUB', 2024);
4. INSERT INTO EDUCATION VALUES (Education_edu_id.NEXTVAL, 'BSc CSE', 'AIUB', 2025);
5. INSERT INTO EDUCATION VALUES (Education_edu_id.NEXTVAL, 'BSc CSE', 'AIUB', 2023);

EDU_ID	DEGREE	INSTITUTION	PASS_YEAR
15	BSc CSE	AIUB	2023
18	BSc CSE	NSU	2024
21	BSc CSE	IUB	2024
24	BSc CSE	AIUB	2025
27	BSc CSE	AIUB	2023

Profile Table:

1. INSERT INTO PROFILE VALUES (Profile_pro_id.NEXTVAL, '', 'ISLAM', 'Engineer', 60, 63, 'hgfdhdgh', 'Cooking', '', 'UNMARRIED', 15, 30);
2. INSERT INTO PROFILE VALUES (Profile_pro_id.NEXTVAL, NULL, 'HINDU', 'Engineer', 66, 52, 'adfadf', 'Guitar', '', 'DIVORCED', 18, 32);
3. INSERT INTO PROFILE VALUES (Profile_pro_id.NEXTVAL, '', 'HINDU', 'Engineer', 63, 70, 'dghdfghd', 'Games', '', 'UNMARRIED', 21, 33);
4. INSERT INTO PROFILE VALUES (Profile_pro_id.NEXTVAL, NULL, 'ISLAM', 'Engineer', 67, 45, 'adfghdfgf', 'Travel', '', 'WIDOWED', 24, 34);
5. INSERT INTO PROFILE VALUES (Profile_pro_id.NEXTVAL, '', 'ISLAM', 'Engineer', 62, 80, 'hsfdghsr', 'Chess', '', 'UNMARRIED', 27, 35);

PROFILE_ID	IMAGE	RELIGION	OCCUPATION	HEIGHT	WEIGHT	BIO	HOBBY1	HOBBY2	MARITAL_STATUS	EDU_ID	USER_ID
84	-	ISLAM	Engineer	60	49	adfadf	Cooking	-	UNMARRIED	15	30
88	-	HINDU	Engineer	66	52	adfadf	Guitar	-	DIVORCED	18	32
92	-	HINDU	Engineer	63	70	dghdfghd	Games	-	UNMARRIED	21	33
96	-	ISLAM	Engineer	67	45	adfghdfgf	Travel	-	WIDOWED	24	34
100	-	ISLAM	Engineer	62	80	hsfdghsr	Chess	-	UNMARRIED	27	35

Preferences Table:

1. INSERT INTO PREFERENCES VALUES (Preferences_pre_id.NEXTVAL, 'ISLAM', 'Engineer', 59, 60, 19, 20, 'Dhaka', '', '', 'UNMARRIED', 84);
2. INSERT INTO PREFERENCES VALUES (Preferences_pre_id.NEXTVAL, 'HINDU', 'Engineer', 62, 62, 19, 21, 'Comilla', '', '', 'DIVORCED', 88);
3. INSERT INTO PREFERENCES VALUES (Preferences_pre_id.NEXTVAL, 'HINDU', 'Engineer', 64, 60, 20, 22, 'Dhaka', '', '', 'UNMARRIED', 92);

4. INSERT INTO PREFERENCES VALUES (Preferences_pre_id.NEXTVAL, 'ISLAM', 'Engineer', 60, 64, 19, 24, 'Khulna', '', '', 'UNMARRIED', 96);
5. INSERT INTO PREFERENCES VALUES (Preferences_pre_id.NEXTVAL, 'ISLAM', 'Engineer', 60, 65, 19, 24, 'Dhaka', '', '', 'UNMARRIED', 100);

PRE_ID	PRE_RELIGION	PRE_OCCUPATION	PRE_HEIGHT	PRE_WEIGHT	AGE_FROM	AGE_TO	PRE_LOCATION	PRE_HOBBY1	PRE_HOBBY2	PRE_MARITAL_STATUS	PROFILE_ID
90	ISLAM	Engineer	59	60	19	20	Dhaka	-	-	UNMARRIED	84
96	HINDU	Engineer	62	62	19	21	Comilla	-	-	DIVORCED	88
102	HINDU	Engineer	64	60	20	22	Dhaka	-	-	UNMARRIED	92
108	ISLAM	Engineer	60	64	19	24	Khulna	-	-	UNMARRIED	96
114	ISLAM	Engineer	60	65	19	24	Dhaka	-	-	UNMARRIED	100

5 rows returned in 0.02 seconds CSV Export

Subscription Table:

1. INSERT INTO SUBSCRIPTION VALUES (Subscription_sub_id.NEXTVAL, 'BRONZE', to_date('12-03-2023','dd-mm-yyyy'), to_date('12-04-2023','dd-mm-yyyy'));
2. INSERT INTO SUBSCRIPTION VALUES (Subscription_sub_id.NEXTVAL, 'SILVER', to_date('10-03-2023','dd-mm-yyyy'), to_date('10-04-2023','dd-mm-yyyy'));
3. INSERT INTO SUBSCRIPTION VALUES (Subscription_sub_id.NEXTVAL, 'GOLD', to_date('07-03-2023','dd-mm-yyyy'), to_date('12-05-2023','dd-mm-yyyy'));
4. INSERT INTO SUBSCRIPTION VALUES (Subscription_sub_id.NEXTVAL, 'BRONZE', to_date('12-03-2023','dd-mm-yyyy'), to_date('12-05-2023','dd-mm-yyyy'));
5. INSERT INTO SUBSCRIPTION VALUES (Subscription_sub_id.NEXTVAL, 'GOLD', to_date('12-03-2023','dd-mm-yyyy'), to_date('12-04-2023','dd-mm-yyyy'));

SUB_ID	TIER	START_DATE	EXPIRE_DATE
2	BRONZE	12-MAR-23	12-APR-23
11	SILVER	10-MAR-23	10-APR-23
20	GOLD	07-MAR-23	12-MAY-23
29	BRONZE	12-MAR-23	12-MAY-23
38	GOLD	12-MAR-23	12-APR-23

Payment Table:

1. INSERT INTO PAYMENT VALUES (Payment_pay_id.NEXTVAL, 'BKASH', 300, to_date('12-03-2023','dd-mm-yyyy'), 'PROCESSED', 30, 2, 50);
2. INSERT INTO PAYMENT VALUES (Payment_pay_id.NEXTVAL, 'NAGAD', 400, to_date('10-03-2023','dd-mm-yyyy'), 'PROCESSED', 32, 11, 60);
3. INSERT INTO PAYMENT VALUES (Payment_pay_id.NEXTVAL, 'BKASH', 500, to_date('07-03-2023','dd-mm-yyyy'), 'PROCESSED', 33, 20, 70);
4. INSERT INTO PAYMENT VALUES (Payment_pay_id.NEXTVAL, 'BANK', 300, to_date('12-03-2023','dd-mm-yyyy'), 'PROCESSED', 34, 29, 80);
5. INSERT INTO PAYMENT VALUES (Payment_pay_id.NEXTVAL, 'BKASH', 500, to_date('12-03-2023','dd-mm-yyyy'), 'PROCESSED', 35, 38, 90);

PAYMENT_ID	METHOD	AMOUNT	PAY_DATE	STATUS	USER_ID	SUB_ID	ADMIN_ID
4	BKASH	300	12-MAR-23	PROCESSED	30	2	50
11	NAGAD	400	10-MAR-23	PROCESSED	32	11	60
18	BKASH	500	07-MAR-23	PROCESSED	33	20	70
25	BANK	300	12-MAR-23	PROCESSED	34	29	80
32	BKASH	500	12-MAR-23	PROCESSED	35	38	90

Conversation Table:

1. INSERT INTO CONVERSATION VALUES (Conversation_con_id.NEXTVAL, 30, 32);
2. INSERT INTO CONVERSATION VALUES (Conversation_con_id.NEXTVAL, 30, 33);
3. INSERT INTO CONVERSATION VALUES (Conversation_con_id.NEXTVAL, 33, 32);
4. INSERT INTO CONVERSATION VALUES (Conversation_con_id.NEXTVAL, 30, 34);
5. INSERT INTO CONVERSATION VALUES (Conversation_con_id.NEXTVAL, 32, 34);

CONVERSATION_ID	USER1	USER2
5	30	32
13	30	33
21	33	32
29	30	34
37	32	34

Messages Table:

1. INSERT INTO MESSAGES VALUES (Messages_msg_id.NEXTVAL, 30, 32, to_date('12-03-2023','dd-mm-yyyy'), 'Hi', 5);
2. INSERT INTO MESSAGES VALUES (Messages_msg_id.NEXTVAL, 32, 30, to_date('12-03-2023','dd-mm-yyyy'), 'Hello', 5);
3. INSERT INTO MESSAGES VALUES (Messages_msg_id.NEXTVAL, 30, 32, to_date('12-03-2023','dd-mm-yyyy'), 'How are you?', 5);
4. INSERT INTO MESSAGES VALUES (Messages_msg_id.NEXTVAL, 32, 30, to_date('12-03-2023','dd-mm-yyyy'), 'Fine, you?', 5);
5. INSERT INTO MESSAGES VALUES (Messages_msg_id.NEXTVAL, 30, 32, to_date('12-03-2023','dd-mm-yyyy'), 'Good', 5);

MSG_ID	SENT_BY	SENT_TO	MSG_DATE	CONTENT	CONVERSATION_ID
110	30	32	12-MAR-23	Hi	5
111	32	30	12-MAR-23	Hello	5
112	30	32	12-MAR-23	How are you?	5
113	32	30	12-MAR-23	Fine, you?	5
114	30	32	12-MAR-23	Good	5

Feedback Table:

1. INSERT INTO FEEDBACK VALUES (Feedback_fd_id.NEXTVAL, 4, 'sdfasd', '', '');
2. INSERT INTO FEEDBACK VALUES (Feedback_fd_id.NEXTVAL, 1, '', 'hshghgh', '');
3. INSERT INTO FEEDBACK VALUES (Feedback_fd_id.NEXTVAL, 2, '', '', 'ghjfghj');
4. INSERT INTO FEEDBACK VALUES (Feedback_fd_id.NEXTVAL, 4, 'sdfasd', '', '');
5. INSERT INTO FEEDBACK VALUES (Feedback_fd_id.NEXTVAL, 4, 'sdfasd', '', '');

FEEDBACK_ID	RATING	R_ACCEPT	R_BLOCK	R_REJECT
5	4	sdfasd	-	-
7	1	-	hshghgh	-
9	2	-	-	ghjfghj
11	4	sdfasd	-	-
13	4	sdfasd	-	-

Match Table:

1. INSERT INTO MATCH VALUES (Match_match_id.NEXTVAL, 30, 32, 80, '', 5, 5);
2. INSERT INTO MATCH VALUES (Match_match_id.NEXTVAL, 30, 33, 75, '', 7, 13);
3. INSERT INTO MATCH VALUES (Match_match_id.NEXTVAL, 33, 32, 33, 'REJECTED', 9, 21);
4. INSERT INTO MATCH VALUES (Match_match_id.NEXTVAL, 30, 34, 90, '', 11, 29);
5. INSERT INTO MATCH VALUES (Match_match_id.NEXTVAL, 32, 34, 64, 'BLOCKED', 13, 37);

MATCH_ID	USER_ID	PARTNER_ID	SCORE	STATUS	FEEDBACK_ID	CONVERSATION_ID
78	30	32	80	-	5	5
87	30	33	75	-	7	13
96	33	32	33	REJECTED	9	21
105	30	34	90	-	11	29
123	32	34	64	BLOCKED	13	37

Query Writing

Single Row Functions:

1. Display name, age and email of all the female users.

SELECT NAME, AGE, EMAIL FROM USERS WHERE UPPER (GENDER) = 'FEMALE'

NAME	AGE	EMAIL
User 1	23	user1@gmail.com
User 2	19	user2@gmail.com
User 4	21	user4@gmail.com

2. Display names and emails of admins whose password length is greater of equal to 4.
   SELECT NAME, EMAIL FROM ADMIN WHERE LENGTH (PASSWORD) >= 4;

NAME	EMAIL
Asif	asif@sfms.com
Siam	siam@sfms.com
Rijoan	rijoan@sfms.com
Fabiha	fabiha@sfms.com

Group Functions:

1. Display the max passing year from each instituition.
   SELECT INSTITUTION, MAX(PASS_YEAR) FROM EDUCATION GROUP BY INSTITUTION

INSTITUTION	MAX(PASS_YEAR)
IUB	2024
NSU	2024
AIUB	2025

2. Display the average age of each gender.
   SELECT GENDER, AVG(AGE) FROM USERS GROUP BY GENDER.

GENDER	AVG(AGE)
Male	21
Female	21

3. Display the minimum height and weight of each marital status group.

SELECT MIN (HEIGHT), MIN (WEIGHT), MARITAL_STATUS FROM PROFILE GROUP BY MARITAL_STATUS;

MIN(HEIGHT)	MIN(WEIGHT)	MARITAL_STATUS
60	49	UNMARRIED
66	52	DIVORCED
67	45	WIDOWED

Subquery:

4. Display the name, age and gender of all users whose age is greater than 'User 3'.
   SELECT NAME, AGE, GENDER FROM USERS WHERE AGE > (SELECT AGE FROM USERS WHERE NAME = 'User 3');

NAME	AGE	GENDER
User 1	23	Female
User 4	21	Female
User 5	22	Male

5. Display Payment method and payment date with greater amounts than payment id 11.
   SELECT METHOD, PAY_DATE FROM PAYMENT WHERE AMOUNT > (SELECT AMOUNT
   FROM PAYMENT WHERE PAYMENT_ID=11);

METHOD	PAY_DATE
BKASH	07-MAR-23
BKASH	12-MAR-23

6. Display Tier and Expire Date of all users whose subscription will expire after the user with sub id 11.
   SELECT TIER, EXPIRE_DATE FROM SUBSCRIPTION WHERE EXPIRE_DATE > (SELECT EXPIRE_DATE
   FROM SUBSCRIPTION WHERE SUB_ID=11);

TIER	EXPIRE_DATE
BRONZE	12-APR-23
GOLD	12-MAY-23
BRONZE	12-MAY-23
GOLD	12-APR-23

Joining:

1. Display RELIGION, OCCUPATION and degree of all the users who has a degree from AIUB.
   SELECT Profile.religion, Profile.occupation, Education.degree FROM PROFILE, EDUCATION WHERE PROFILE.EDU_ID = EDUCATION.EDU_ID AND EDUCATION.INSTITUTION = 'AIUB';

RELIGION	OCCUPATION	DEGREE
ISLAM	Engineer	BSc CSE
ISLAM	Engineer	BSc CSE
ISLAM	Engineer	BSc CSE

2. Display Name, Email and Zip Code of all users who lives in Dhaka City.
   SELECT Users.name, Users.email, Address.zip_code FROM Users, Address WHERE Users.add_id = Address.add_id AND Address.city = 'DHAKA';

NAME	EMAIL	ZIP_CODE
User 1	user1@gmail.com	1215
User 5	user5@gmail.com	1659

3. Display Height, Weight and Location Preferences of all users whose weight is above 65.
   SELECT Preferences.pre_weight, Preferences.pre_height, Preferences.pre_location FROM Profile, Preferences WHERE Profile.profile_id = Preferences.profile_id AND Profile.weight > 65;

PRE_WEIGHT	PRE_HEIGHT	PRE_LOCATION
60	64	Dhaka
65	60	Dhaka

Views:

1. Create a view called UsersView based on the user name, user age and user email from Users table.
   CREATE VIEW UsersView AS SELECT name, age, email FROM Users;
   SELECT * FROM UsersView;

NAME	AGE	EMAIL
User 1	23	user1@gmail.com
User 2	19	user2@gmail.com
User 3	20	user3@gmail.com
User 4	21	user4@gmail.com
User 5	22	user5@gmail.com

2. Write a query to display all users name and email from UsersView whose age is below 22.
   SELECT name, email FROM UsersView WHERE age < 22;

NAME	EMAIL
User 2	user2@gmail.com
User 3	user3@gmail.com
User 4	user4@gmail.com

3. Create a view called ProfileView based on profile id, religion and occupation from Profile table.
   CREATE VIEW ProfileView AS SELECT profile_id, religion, occupation FROM Profile;
   SELECT * FROM ProfileView;

PROFILE_ID	RELIGION	OCCUPATION
84	ISLAM	Engineer
88	HINDU	Engineer
92	HINDU	Engineer
96	ISLAM	Engineer
100	ISLAM	Engineer

Relational Algebra

1. Find the name of the user whose user id is 33.

   $\Pi_{name}(\sigma_{user\_id = 33}(Users))$

2. Find the age of 'User 1'.

   $\Pi_{age}(\sigma_{name = \text{"User 1"}}(Users))$

3. Find the house no, street name and postal code where add_id is equal to 12.

   $\Pi_{house\_no,\ street\_no,\ zip\_code,\ city}(\sigma_{add\_id = 12}(Address))$

4. Find user ids which birthday is on 2000-01-01.

   $\Pi_{user\_id}(\sigma_{dob = \text{"2000-01-01"}}(Users))$

5. Find the education Id whose passing year is after 2022.

   $\Pi_{edu\_id}(\sigma_{pass\_year > 2022}(Education))$

Conclusion

Conclusion:

In this project, we have mainly created a sql-based spouse finding management system focusing on perfect match in an error-free manner to reduce all sorts of hassle that happen in a traditional management system. The project queries run in 'Oracle 10g'. Here, we have made many relationship among all the entities with cardinality. We have also normalized the project to organize data elements properly.

Future planning:

Future scope of spouse finding system project is to provide a platform to a lot of Bride/Groom for finding perfect matches. There are different sectors such as Registration, Partner, Search, etc. In such a way the bride/groom will get their attention for finding their partner. The Bride/Groom can directly search for a Partner according to the criteria they require. The Bride/Groom can use the match by email capability so they will receive an Email alert when a match fulfills their criteria.

Since Bangladesh is a country of a large population, we have many religions, and each religion is further divided into caste, which is further divided into sub caste, etc. so this kind of application would be helpful for each group.

Through this application, users can also narrow their search for partners to those who meet a certain set of criteria for online marriage. Using the Internet as a pivot for modern business, our project, based on the internet, creates a pathway for modernizing trade.

Most people used to hesitate sending photos of arranged marriage proposals, and even when they did, there was always a problem with the Here, you can just shortlist the profiles with photos only.

Web Application:

Spouse Finding web Application is aimed at providing Grooms and Brides with an extraordinary matchmaking experience by providing resources and opportunities for meeting a true potential partner. This site provides users with an overview of how to commit on the website. Marriage Web applications provide the facility to change preference about partners. The user can edit his profile, update his picture, delete his picture, hide his profile, create an album, send a personal message etc.