

Project-1

Team Members (Team 5):

- 1) Sampada Dhole (NUID: 001526241)
- 2) Meenakshi Vijay Kumbhar (NUID: 002922024)
- 3) Vishnuprasad Thopucharla (NUID: 002926400)
- 4) Nikhitha Dasi Srinivas (NUID: 002922544)
- 5) Shravya Vura (NUID: 002927192)

Project Topic

University students social networking database

Problem statement

Due to the pandemic, It has become difficult for students to interact with their colleagues. As the teaching method is changed from traditional on ground classes to Online classes, many students are not getting the time and chance to interact with each other. "Social Network" is omnipresent in today's world, even though there are many social networking sites where people can communicate with each other, the only factor missing is the security. While networking, we come across fake profiles due to which people get affected personally and professionally.

The idea behind this project is to develop verified student network with other graduate students. So that it gets easier for students to communicate based on the coursework they have opted for.

Objective

This project aims to help students to create a network with other students depending on the course selected, college wise. This will help them to have a wide range of student networks who can help each other for the betterment of their career growth. Many students feel that using social networking to enhance their personal learning network and to discover resources is very beneficial. Almost every student uses social networking in some form as their daily routine.

Considering security the major concern for University students, we are aiming to design our project to resolve the security issues caused mainly due to fake identification of profiles and provide a safe and interactive environment for students to network and discuss both course related topics and build networking based on personal interests.

PROJECT 2

University students social networking database

Sampada Dhole (NUID: **001526241**)

Meenakshi Vijay Kumbhar (NUID: **002922024**)

Vishnuprasad Thopucharla (NUID: **002926400**)

Nikhitha Dasi Srinivas (NUID: **002922544**)

Shravya Vura (NUID: **002927192**)

As social networking becomes more popular, a variety of things are changing, from how we get information to how we communicate and, most importantly, the way we interact. We cannot ignore how social media outlets are becoming more important and relevant as they develop. The obvious perk of social networking is interconnecting with one another, and social networking sites create a repository of personal information. Unsurprisingly, social networks are not immune to security risks. A significant portion of social media is about privacy. The problem is that teenagers, who are less concerned about privacy and security, continue to give up information freely. This is one of the main reasons why teenagers' information is being compromised.

Because of the pandemic, student interactions have been hindered. In addition, since classes have changed from on-ground to online, many students are not having the opportunity to interact with their classmates. Also, many students feel that using social networking to enhance their personal learning network and to discover resources is very beneficial. Almost every student uses social networking in some form as their daily routine. Even though social networking is omnipresent in today's world, security is a big concern. We come across fake profiles when networking, which is how we are personally and professionally impacted.

Goals of the project:

- Establish a **verified student network database** with other **graduate students**, so they can get in touch based on the coursework they have chosen and have a wide range of student networks who can help each other for the **betterment of their career growth**.
- **Resolve the security issues** caused mainly due to **fake identification** of profiles and provide a safe and interactive environment for students to network and discuss both course related topics and build networking based on personal interests.

Only the student's whose OTP gets verified will be able to login to the application.

Once the student is logged into the system, his personal details along with the student's University, Program and Course details will be stored in the **SNDB_USER_ACCOUNT** table, which can be viewed and modified by the user.

Each time a student logs in, student's login history will be stored in the **SNDB_LOGGED_IN_DATA** table.

SNDB_GENDER_DATA table will contain the list of genders available in the database.

Students can upload their profile photo and the photo details and visibility details are maintained in the **SNDB_USER_PHOTO_DATA** table.

Students also have the provision to post multiple posts and the respective post data is stored in the **SNDB_POST_DATA** table. The **SNDB_VOTES_DATA** will contain the number of likes and dislikes of the post. This information can be only viewed by the user.

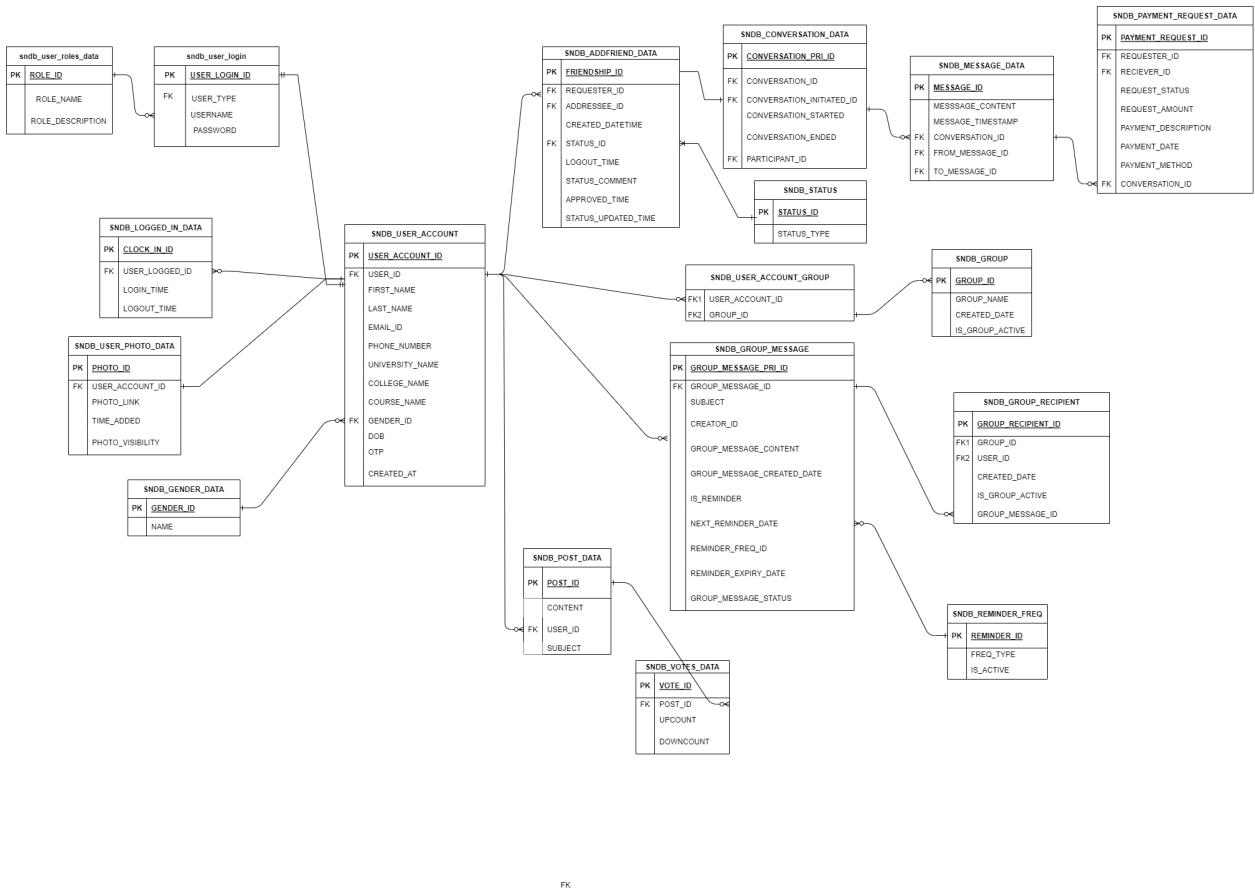
Students can send multiple friend requests and the respective request and status details are stored in **SNDB_ADDFRIEND_DATA** and **SNDB_STATUS** table

Students can start a conversation with multiple people and send messages, the conversation details and message details are stored in **SNDB_CONVERSATION_DATA** and **SNDB_MESSAGE_DATA** tables. This can be viewed only by the respective students currently utilizing the chat feature.

During the conversation, students can request for money transfer and the payment request and status details are stored in the **SNDB_PAYMENT_REQUEST_DATA** table.

Students can also join multiple groups and send group messages; these details are maintained in **SNDB_USER_ACCOUNT_GROUP**, **SNDB_GROUP**, **SNDB_GROUP_RECIPIENT**, and **SNDB_GROUP_MESSAGE** tables.

During the group messages, students are given an option to send reminder messages in the chat, the reminder message details are stored in the **SNDB_GROUP_MESSAGE** table and the **SNDB_Reminder_FREQ** table contains the list of available frequencies for sending reminders.



Relationships in ER Diagram- For reference

Analogies : > many-to-one; < one-to-many; - one-to-one

Ref:SNDB_USER_ROLES_DATA.role_name<SNDB_USER_LOGIN.user_type
 Ref:SNDB_LOGGED_IN_DATA.USER_LOGGED_ID-SNDB_USER_ACCOUNT.user_id
 Ref:SNDB_GENDER_DATA.gender_id < SNDB_USER_ACCOUNT.gender_id
 Ref: SNDB_USER_ACCOUNT.user_id < SNDB_POST_DATA.user_id
 Ref:SNDB_USER_PHOTO_DATA.user_account_id - SNDB_USER_ACCOUNT.user_id
 Ref:SNDB_POST_DATA.post_id < SNDB_VOTES_DATA.post_id
 Ref: SNDB_USER_ACCOUNT.user_id < SNDB_ADDFRIEND_DATA.requester_id
 Ref: SNDB_USER_ACCOUNT.user_id < SNDB_ADDFRIEND_DATA.addressee_id
 Ref: SNDB_ADDFRIEND_DATA.FRIENDSHIP_ID - SNDB_CONVERSATION_DATA.CONVERSATION_ID
 Ref:
 SNDB_CONVERSATION_DATA.CONVERSATION_INITIATED_ID>SNDB_USER_ACCOUNT.USER_ID
 Ref: SNDB_CONVERSATION_DATA.PARTICIPANT_ID > SNDB_USER_ACCOUNT.USER_ID
 Ref: SNDB_STATUS.STATUS_ID > SNDB_ADDFRIEND_DATA.STATUS_ID
 Ref:
 SNDB_CONVERSATION_DATA.CONVERSATION_ID-SNDB_MESSAGE_DATA.CONVERSATION_ID

Ref: SNDB_MESSAGE_DATA.CONVERSATION_ID <
 SNDB_PAYMENT_REQUEST_DATA.CONVERSATION_ID
 Ref:SNDB_USER_ACCOUNT.USER_ID < SNDB_USER_ACCOUNT_GROUP.USER_ACCOUNT_ID
 Ref: SNDB_USER_ACCOUNT_GROUP.USER_ACCOUNT_ID < SNDB_GROUP.GROUP_ID
 Ref:SNDB_USER_ACCOUNT.USER_ID < SNDB_GROUP_MESSAGE.CREATOR.ID
 Ref:SNDB_GROUP_MESSAGE.GROUP_MESSAGE_ID <
 SNDB_GROUP_RECIPIENT.GROUP_MESSAGE_ID
 Ref:SNDB_Reminder_FREQ.Reminder_ID <
 SNDB_GROUP_MESSAGE.Reminder_FREQ_ID

student_account	
student_id	int
first_name	varchar2
last_name	varchar2
email_id	varchar2
phone_number	int
university_name	int
college_name	int
course_name	int
gender_id	int
dob	date
otp	int
created_at	timestamp
student_type	varchar2

student_photo_data	
photo_id	int
student_account_id	int
photo_link	varchar2
time_added	timestamp
photo_visibility	varchar2

post_data	
post_id	int
subject	varchar2
content	varchar2
student_id	int

message_data	
message_id	int
message_content	varchar2
message_timestamp	timestamp
conversation_id	int
from_message_id	int
to_message_id	int

conversation_data	
conversation_id	int
conversation_initiated_id	int
conversation_started	timestamp
conversation_ended	timestamp
participant_id	int

votes_data	
vote_id	int
post_id	int
upcount	int
downcount	int

status	
status_id	int
status_type	varchar2

gender_data	
gender_id	int
name	varchar2

payment_request_data	
payment_request_id	int
requester_id	int
reciever_id	int
request_status	varchar2
request_amount	int
payment_description	varchar2
payment_date	date
payment_method	varchar2
conversation_id	int

group	
group_id	int
group_name	varchar2
created_date	date
is_group_active	varchar2

student_account_group	
student_group_id	int
student_account_id	int
group_id	int

group_recipient	
group_recipient_id	int
group_id	int
created_date	date
is_group_active	varchar2
group_message_id	int
is_read	int

group_message	
group_message_id	int
subject	varchar2
creator_id	int
group_message_content	clob
group_message_created_date	date
is_reminder	varchar2
nextReminderDate	date
reminder_freq_id	int
reminder_expiry_date	date

reminder_freq	
reminder_id	int
freq_type	varchar2
is_active	varchar2

logged_in_data	
clock_in_id	int
student_logged_id	int
login_time	timestamp
logout_time	timestamp

addfriend	
friendship_id	int
requester_id	int
addressee_id	int
created_datetime	timestamp
status_id	int
status_comment	varchar2
approved_time	timestamp

a PROJECT 3 (Team 5)

University Students Social Networking Database

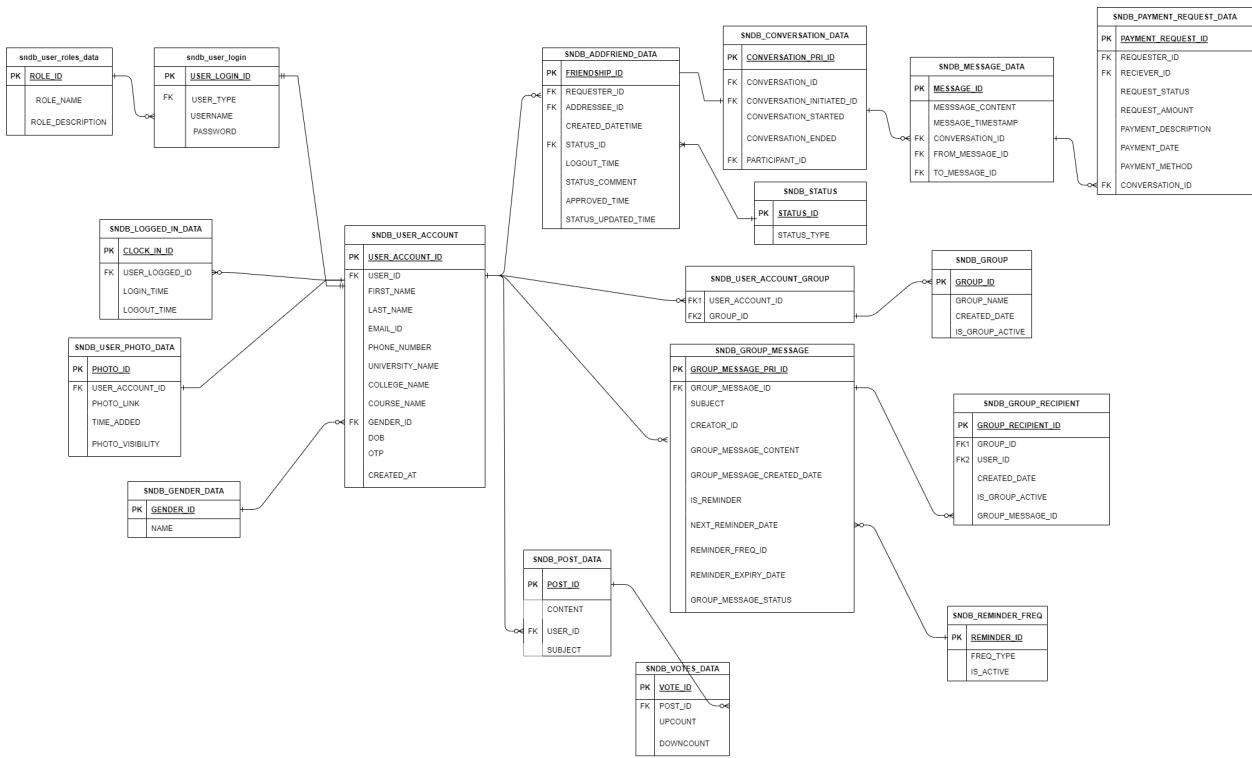
Sampada Dhole (NUID: **001526241**)
Meenakshi Vijay Kumbhar (NUID: **002922024**)
Vishnuprasad Thopucharla (NUID: **002926400**)
Nikhitha Dasi Srinivas (NUID: **002922544**)
Shravya Vura (NUID: **002927192**)

Business Rules:

Entity	Rules
User login	<ol style="list-style-type: none">1) Admin or student can login using their username and password2) User must have one role, either student or admin
User Account	<ol style="list-style-type: none">1) User account should add their personal details like First Name, Last Name, Phone Number, Email Address2) Student needs to be enrolled in a university and must provide university Name, College Name, Course Name3) User can be uniquely identified by user_id4) All the students in the user_account database are considered as verified students.
Post Data	<ol style="list-style-type: none">1) Student can only post in text format and the data will be stored in the post_data.2) Students can see post even if they are not friends3) Students can dislike or like the posts
Add Friend	<ol style="list-style-type: none">1) Based on the status, the student can be added to the friend list, unfriend the student.2) Only after accepting the friend request, one student can have conservation with each other
Votes data	<ol style="list-style-type: none">1) Student can see the upcount and downcount values of the post2) Students can vote posts of other students even if they are not friends
Message Data	<ol style="list-style-type: none">1) One on one conversation data will be stored in this.

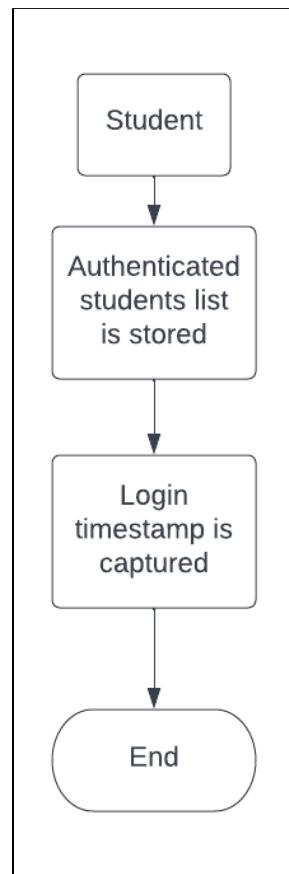
	<p>2) Message data will be identified by the primary key message_id</p>
Payment Request Data	<p>1) If the payment request is accepted, bill details are displayed.</p>
Group message data	<p>1) Group conversation can only be started with a minimum of 3 group members. 2) Group members can send reminder messages to the group with the expiry date of the reminder.</p>
Group Recipient Data	<p>1) Group will deactivate when the is_group_active flag is set to false. 2) Group creator's data will be stored in this.</p>
Reminder Frequency Data	<p>1) When the is_reminder flag is set to yes, then the message reminder data is sent to the group. 2) Group message reminders can be based on the reminder frequency such as weekly, daily and monthly.</p>
Photo data	<p>1) Student can view/update their own profile photo.</p>

Revised ER diagram:

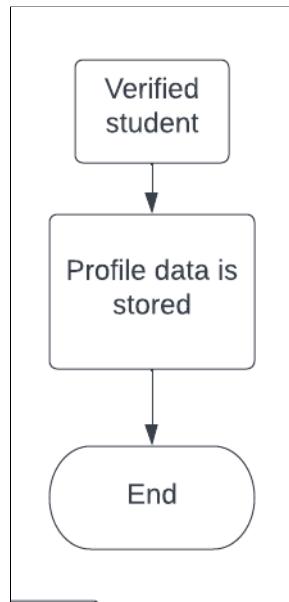


DFD Diagram:

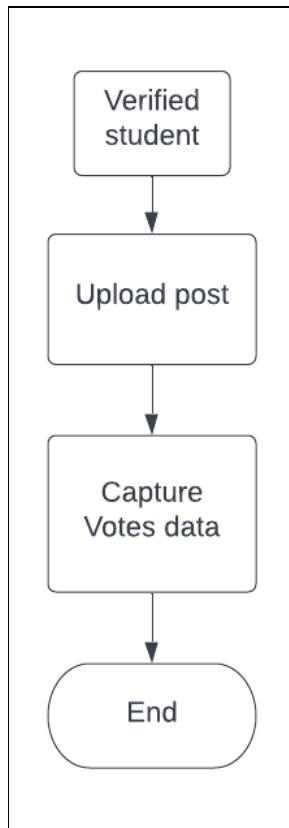
1. User Account Login Flow:



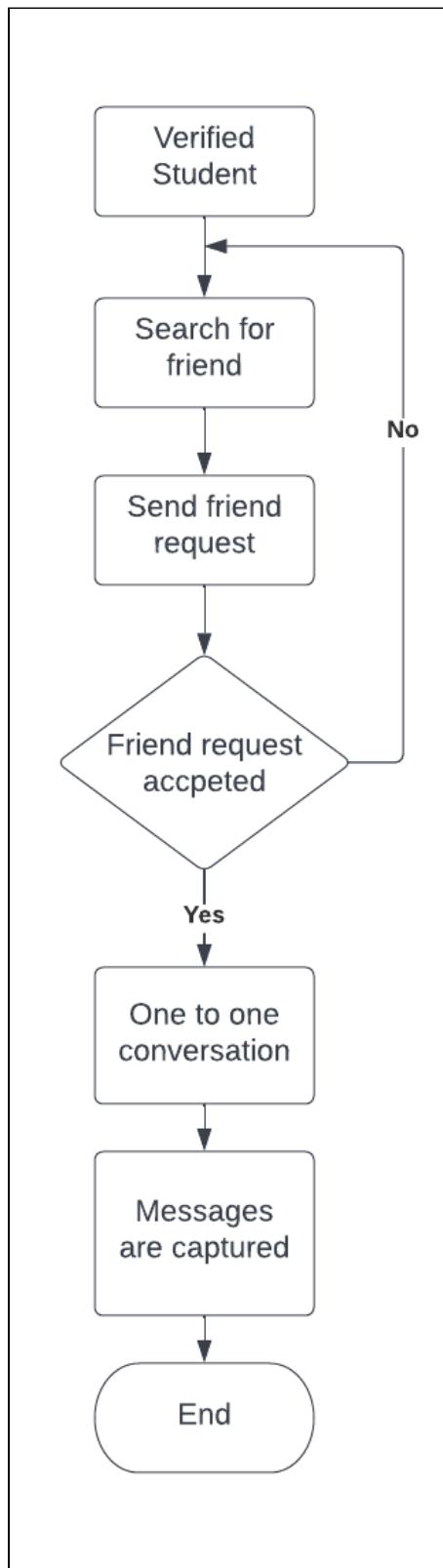
2. Profile Picture Flow



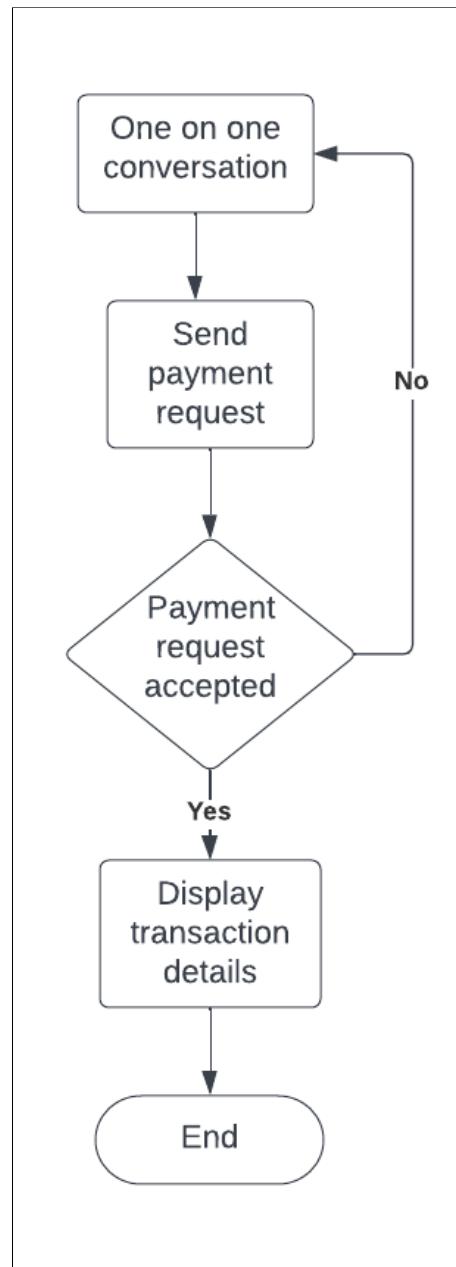
3. Post and Votes Flow



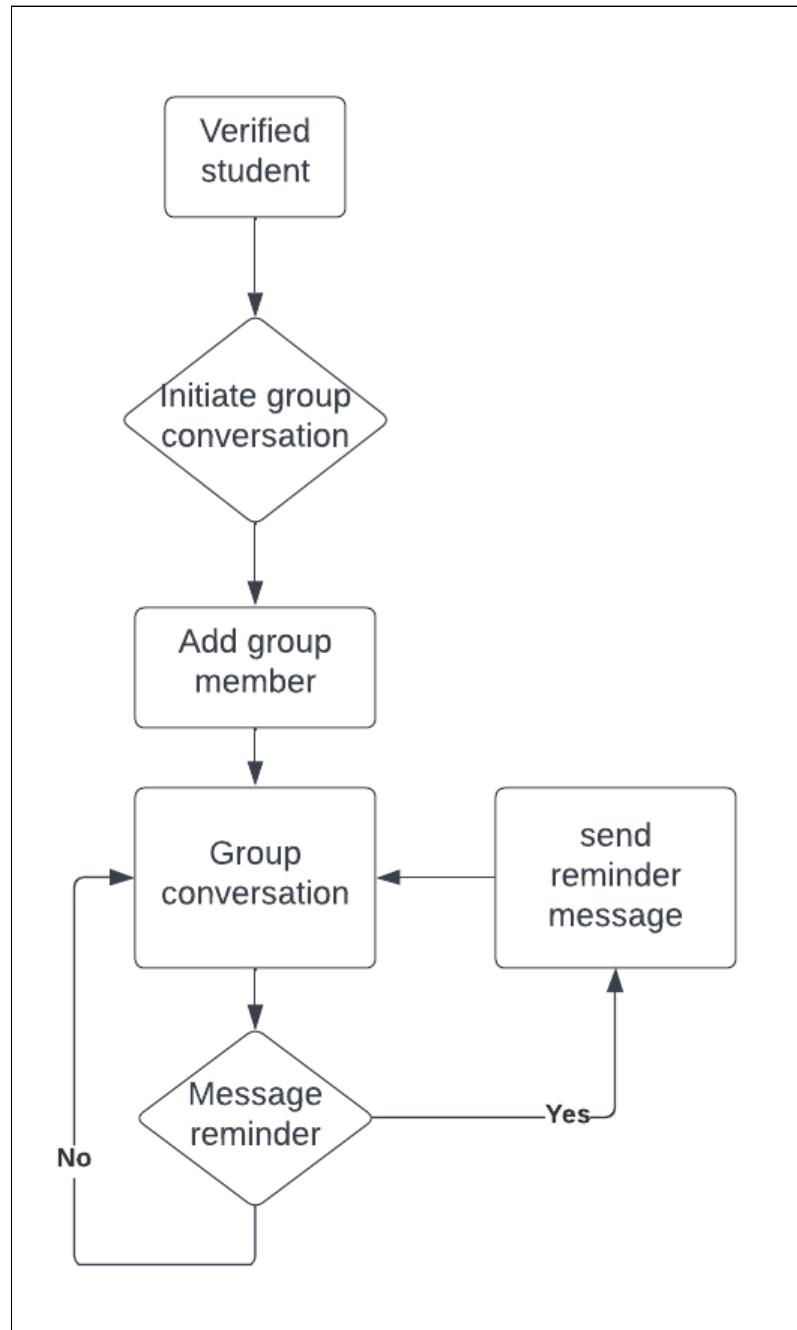
4. Friend Request and One on One Conversation Flow:



5. Payment Control Flow



6. Group Message Flow:



VIEWS:

Role: Admin		
View Name	Access	Description/Example
all_data	View Data/Update/Modify table properties/Insert/Delete	Admin has access to all views and tables
View_account_details	View Data	View Data - profile details
View_login_usertype_admin_details	View Data/Update Data	Admin can view/update own login details ie view username, update password

Role: Student		
View Name	Access	Description/Example
View_Student_details	View/update Data	Student can view/update own profile data
View_Student_profile_photo	View/Update Data	Student can view own profile photo.

View_Student_Post_Details	View/update Data	Student can view post details
View_Student_Friends_Details	View Data	Student can view friends profile data and messages
View_Student_One_on_One_Conversations	View/Update Data	Student can view their Conversations with friends
View_Student_Payment_Details	View Data	Student can view the Payment request details
View_Student_Group_Messages	View/update Data	Students can view the group messages and update reminder frequencies.

SECURITY:

Access Control	
Admin Level Access	Student Level Access
sndb_user_account can be accessed by admin.	sndb_addfriend: Student can access their own friends data
sndb_logged_in_data can be accessed by admin.	sndb_post_data can be accessed by student.
sndb_addfriend can be accessed by admin	sndb_group can be accessed by student.
sndb_status can be accessed by admin.	sndb_user_photo_data can be accessed by student to update/view their own Photo
sndb_gender_data can be accessed by only admin.	sndb_user_account can be accessed by student to view/update their own Account data.
sndb_post_data can be accessed by admin.	sndb_votes_data can be accessed by student.
sndb_votes_data can be accessed by admin.	sndb_payment_request_data can be accessed by student to view their own conversation Payment request data
sndb_payment_request_data can be accessed by admin.	sndb_message_data can be accessed by student to send messages to their friend list
sndb_user_photo_data can be accessed by admin.	sndb_group_message can be accessed by student to send group messages.
sndb_conversation_data can be accessed by admin.	
sndb_message_data can be accessed by both admin.	
sndb_user_account_group can be accessed by admin.	
sndb_group can be accessed by admin.	
sndb_group_message can be accessed by admin.	

group_recipient can be accessed by admin.	
--	--

PROJECT 4 (Team 5)

Script Code:

```
SET SERVEROUTPUT ON;
```

```
begin
execute immediate 'DROP TABLE SNDB_group_message';
exception when others then null;
end;
/
```

```
begin
execute immediate 'DROP TABLE SNDB_reminder_freq';
exception when others then null;
end;
/
```

```
begin
execute immediate 'alter table SNDB_group_recipient drop constraint FK_user_grp_ID_key';
exception when others then null;
end;
/
```

```
begin
execute immediate 'drop table SNDB_group_recipient';
exception when others then null;
end;
/
```

```
begin
execute immediate 'alter table SNDB_user_account_group drop constraint user_group_ID';
exception when others then null;
end;
/
```

```
begin
execute immediate 'drop table SNDB_user_account_group';
exception when others then null;
end;
/
```

```
begin
execute immediate 'drop table SNDB_group';
exception when others then null;
end;
/

begin
execute immediate 'alter table SNDB_payment_request_data drop constraint
FK_conversation_id';
exception when others then null;
end;
/

begin
execute immediate 'alter table SNDB_payment_request_data drop constraint
Fk_paymnt_requester_id';
exception when others then null;
end;
/

begin
execute immediate 'alter table SNDB_payment_request_data drop constraint
Fk_paymnt_reciever_id';
exception when others then null;
end;
/

begin
execute immediate 'Drop table SNDB_payment_request_data';
exception when others then null;
end;
/

begin
execute immediate 'alter table SNDB_message_data drop constraint UQ_conversation_id';
exception when others then null;
end;
/
begin
execute immediate 'alter table SNDB_message_data drop constraint Fk_To_messg_id';
exception when others then null;
end;
/
```

```
begin
execute immediate 'alter table SNDB_message_data drop constraint Fk_From_messg_id';
exception when others then null;
end;
/

begin
execute immediate 'Drop table SNDB_message_data';
exception when others then null;
end;
/

begin
execute immediate 'alter table SNDB_conversation_data drop constraint Fk_frd_user_start_id';
exception when others then null;
end;
/
begin
execute immediate 'alter table SNDB_conversation_data drop constraint FK_participant_id';
exception when others then null;
end;
/
begin
execute immediate 'DROP TABLE SNDB_conversation_data';
exception when others then null;
end;
/
begin
execute immediate 'alter table sndb_addfriend_data drop constraint Fk_friend_status_id';
exception when others then null;
end;
/
begin
execute immediate 'DROP TABLE sndb_addfriend_data';
exception when others then null;
end;
/
begin
execute immediate 'alter table SNDB_STATUS drop constraint CHK_STATUS';
exception when others then null;
```

```
end;
/

begin
execute immediate 'Drop table SNDB_STATUS';
exception when others then null;
end;
/

begin
execute immediate 'Drop table SNDB_user_photo_data';
exception when others then null;
end;
/

begin
execute immediate 'Drop table SNDB_LOGGED_IN_DATA';
exception when others then null;
end;
/

begin
execute immediate 'Drop table SNDB_VOTES_DATA';
exception when others then null;
end;
/

begin
execute immediate 'Drop table SNDB_post_data';
exception when others then null;
end;
/

begin
execute immediate 'alter table SNDB_USER_ACCOUNT drop constraint UQ_fields';
exception when others then null;
end;
/

begin
execute immediate 'Drop table SNDB_USER_ACCOUNT';
exception when others then null;
end;
/

begin
execute immediate 'drop table SNDB_gender_data';
exception when others then null;
```

```

end;
/
begin
execute immediate 'Drop table SNDB_user_login';
exception when others then null;
end;
/
begin
execute immediate 'Drop table SNDB_user_roles_data';
exception when others then null;
end;
/

--USER_ROLES data Table creation--
DECLARE
    p_user_roles_data NUMBER;
    user_roles_data_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_user_roles_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_user_roles_data';

    IF p_user_roles_data = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_user_roles_data(
            role_id number NOT NULL PRIMARY KEY,
            role_name varchar2(50) NOT NULL unique,
            role_description varchar2(50)

        )';
        DBMS_OUTPUT.PUT_LINE('TABLE CONFIG_TABLE CREATED SUCCESSFULLY');
    ELSE
        RAISE user_roles_data_table_exsists;
    END IF;
EXCEPTION
    WHEN user_roles_data_table_exsists THEN
        dbms_output.put_line('SNDB_user_roles_data Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_user_roles_data Table already in use by existing object');
        END IF;
END;

```

```

        END IF;
END;
/

DECLARE
    p_user_login NUMBER;
    user_login_exists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_user_login
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_user_login';

    IF p_user_login = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_user_login(
            user_login_id number NOT NULL,
            username varchar2(50) NOT NULL,
            password varchar2(50) NOT NULL,
            user_type varchar2(50) not null,
            CONSTRAINT user_login_id_pk PRIMARY KEY (user_login_id),
            CONSTRAINT FK_user_type FOREIGN KEY (user_type)
                REFERENCES SNDB_USER_ROLES_DATA(ROLE_NAME)
        )';
    ELSE
        RAISE user_login_exists;
    END IF;
EXCEPTION
    WHEN user_login_exists THEN
        dbms_output.put_line('SNDB_user_login Table already exists');
        WHEN OTHERS THEN
            IF SQLCODE = -955 THEN
                dbms_output.put_line('SNDB_user_login Table already in use by existing object');
            END IF;
    END;
/
--gender_data data Table creation--

DECLARE
    p_gender_data NUMBER;
    gender_data_table_exists EXCEPTION;

```

```

BEGIN
  SELECT
    COUNT(*)
  INTO p_gender_data
  FROM
    user_tables
  WHERE
    table_name = 'SNDB_GENDER_DATA';

  IF p_gender_data = 0 THEN
    EXECUTE IMMEDIATE 'CREATE TABLE SNDB_gender_data (
      gender_id number NOT NULL,
      name varchar2(50) NOT NULL,
      CONSTRAINT gender_id_pk PRIMARY KEY (gender_id ) )';
  ELSE
    RAISE gender_data_table_exists ;
  END IF;
EXCEPTION
  WHEN gender_data_table_exists THEN
    dbms_output.put_line('SNDB_gender_data Table already exists');
  WHEN OTHERS THEN
    IF SQLCODE = -955 THEN
      dbms_output.put_line('SNDB_gender_data Table already in use by existing object');
    END IF;
END;
/

```

--USER_ACCOUNT Table creation--

```

DECLARE
  p_user_account NUMBER;
  user_account_table_exists EXCEPTION;
BEGIN
  SELECT
    COUNT(*)
  INTO p_user_account
  FROM
    user_tables
  WHERE
    table_name = 'SNDB_user_account';

  IF p_user_account = 0 THEN
    EXECUTE IMMEDIATE 'CREATE TABLE SNDB_user_account(

```

```

        user_account_id number not null,
        user_id int not null unique,
        first_name varchar2(50) not null,
        last_name varchar2(50) not null,
        email_id varchar2(50) not null,
        phone_number int,
        university_name varchar2(50),
        college_name varchar2(50),
        course_name varchar2(50),
        gender_id int,
        dob date,
        otp int,
        created_at timestamp,
        CONSTRAINT user_id_pk PRIMARY KEY (user_account_id),
        CONSTRAINT FK_user_id FOREIGN KEY (gender_id)
        REFERENCES SNDB_gender_data(gender_id)
    );
    ELSE
        RAISE user_account_table_exsists ;
    END IF;
EXCEPTION
    WHEN user_account_table_exsists THEN
        dbms_output.put_line('SNDB_user_account Table already exists');
        WHEN OTHERS THEN
            IF SQLCODE = -955 THEN
                dbms_output.put_line('SNDB_user_account Table already in use by existing object');
            END IF;
    END;
/
ALTER TABLE SNDB_USER_ACCOUNT
ADD CONSTRAINT UQ_fields UNIQUE (email_id, phone_number);

ALTER TABLE SNDB_USER_ACCOUNT
MODIFY otp int NOT NULL;

ALTER TABLE SNDB_USER_ACCOUNT
ADD CONSTRAINT FK_user_login_id
FOREIGN KEY (user_id) REFERENCES SNDB_USER_LOGIN(user_login_id);

DECLARE
    p_post_data_query NUMBER;
    post_data_table_exsists EXCEPTION;
BEGIN
    SELECT

```

```

        COUNT(*)
INTO p_post_data_query
FROM
    user_tables
WHERE
    table_name = 'SNDB_post_data';

IF p_post_data_query = 0 THEN
    EXECUTE IMMEDIATE 'CREATE TABLE SNDB_POST_DATA(
post_id number NOT NULL,
subject varchar2(50) NOT NULL,
content varchar2(50),
user_id number NOT NULL,
CONSTRAINT post_id_pk PRIMARY KEY (post_id),
CONSTRAINT FK_POST_user_id FOREIGN KEY (user_id)
REFERENCES SNDB_USER_ACCOUNT(user_id)
)';
ELSE
    RAISE post_data_table_exsists;
END IF;
EXCEPTION
    WHEN post_data_table_exsists THEN
        dbms_output.put_line('SNDB_post_data Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_post_data Table already in use by existing object');
        END IF;
END;
/

```

--Votes data Table creation--

```

DECLARE
    p_votes_data NUMBER;
    votes_data_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_votes_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_VOTES_DATA';

```

```

IF p_votes_data = 0 THEN
    EXECUTE IMMEDIATE 'CREATE TABLE SNDB_votes_data(
        vote_id number NOT NULL,
        post_id number NOT NULL,
        upcount number,
        downcount number,
        CONSTRAINT vote_id_pk PRIMARY KEY (vote_id),
        CONSTRAINT FK_post_id FOREIGN KEY (post_id)
            REFERENCES SNDB_POST_DATA(post_id)
    )';

ELSE
    RAISE votes_data_table_exsists;
END IF;

EXCEPTION
    WHEN votes_data_table_exsists THEN
        dbms_output.put_line('SNDB_votes_data Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_votes_data Table already in use by existing object');
        END IF;
END;
/

```

--Logged_in data Table creation--

```

DECLARE
    p_loggedin_data NUMBER;
    logged_in_data_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_loggedin_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_LOGGED_IN_DATA';

    IF p_loggedin_data = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_logged_in_data(
            clock_in_id number NOT NULL,
            user_logged_id number NOT NULL,
            login_time timestamp,

```

```

        logout_time timestamp,
        CONSTRAINT clock_in_id_pk PRIMARY KEY (clock_in_id),
        CONSTRAINT FK_user_logged_id FOREIGN KEY (user_logged_id)
        REFERENCES SNDB_USER_ACCOUNT(user_id)

    )';

ELSE
    RAISE logged_in_data_table_exsists;
END IF;
EXCEPTION
    WHEN logged_in_data_table_exsists THEN
        dbms_output.put_line('SNDB_logged_in_data Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_logged_in_data Table already in use by existing object');
        END IF;
END;
/

```

--USER_PHOTO DATA TABLE CREATION

```

DECLARE
    p_user_photo_data NUMBER;
    user_photo_data_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_user_photo_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_user_photo_data';

    IF p_user_photo_data = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_user_photo_data(
            photo_id number NOT NULL ,
            user_account_id number NOT NULL,
            photo_link varchar2(50),
            time_added timestamp,
            photo_visibility varchar2(10),
            CONSTRAINT photo_id_pk PRIMARY KEY (photo_id ),
            CONSTRAINT FK_user_account_id FOREIGN KEY (user_account_id )

```

```

    REFERENCES SNDB_user_account(user_id)
);
ELSE
    RAISE user_photo_data_table_exists ;
END IF;
EXCEPTION
    WHEN user_photo_data_table_exists THEN
        dbms_output.put_line('SNDB_user_photo_data Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_user_photo_data Table already in use by existing object');
        END IF;
END;
/

```

--status Table creation--

```

DECLARE
    p_status NUMBER;
    status_table_exists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_status
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_STATUS';
    IF p_status = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_Status(
            status_id number NOT NULL,
            status_type varchar2(30) NOT NULL,
            CONSTRAINT status_id_pk PRIMARY KEY (status_id)

)';
    ELSE
        RAISE status_table_exists;
    END IF;
EXCEPTION
    WHEN status_table_exists THEN
        dbms_output.put_line('SNDB_Status Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_Status Table already in use by existing object');
        END IF;

```

```

        END IF;
END;
/

ALTER TABLE SNDB_STATUS
ADD CONSTRAINT CHK_STATUS CHECK (STATUS_TYPE
IN('Accepted','Rejected','Pending'));

--addfriend data Table creation--

DECLARE
    p_addfriend_data NUMBER;
    addfriend_data_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_addfriend_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_addfriend_data';

    IF p_addfriend_data = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_addfriend_data(
            friendship_id number NOT NULL,
            requester_id number NOT NULL,
            addressee_id number NOT NULL,
            created_datetime timestamp,
            status_id number NOT NULL,
            logout_time timestamp,
            Status_comment varchar2(50),
            approved_time timestamp,
            status_updated_time timestamp,
            CONSTRAINT friendship_id_pk PRIMARY KEY (friendship_id ),
            CONSTRAINT FK_requester_id FOREIGN KEY (requester_id)
            REFERENCES SNDB_user_account(user_id),
            CONSTRAINT FK_addressee_id FOREIGN KEY (addressee_id)
            REFERENCES SNDB_user_account(user_id)
        )';

        ELSE
            RAISE addfriend_data_table_exsists ;
        END IF;
    END;

```

```

EXCEPTION
WHEN addfriend_data_table_exsists THEN
    dbms_output.put_line('SNDB_addfriend_data Table already exists');
WHEN OTHERS THEN
    IF SQLCODE = -955 THEN
        dbms_output.put_line('SNDB_addfriend_data Table already in use by existing object');
    END IF;
END;
/
ALTER TABLE sndb_addfriend_data
ADD CONSTRAINT Fk_friend_status_id
    FOREIGN KEY (status_id)
    REFERENCES sndb_STATUS (status_id );

-- conversation_data Table creation–

DECLARE
    p_conversation_data NUMBER;
    conversation_data_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_conversation_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_CONVERSATION_DATA';

    IF p_conversation_data = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_conversation_data(
            conversation_pri_id number NOT NULL,
            conversation_id number NOT NULL,
            conversation_initiated_id number NOT NULL,
            conversation_started timestamp,
            conversation_ended timestamp,
            participant_id number NOT NULL,
            CONSTRAINT convers_pri_id_pk PRIMARY KEY (conversation_pri_id),
            CONSTRAINT frd_conversation_id_FK FOREIGN KEY (conversation_id)
                REFERENCES SNDB_ADDFRIEND_DATA(FRIENDSHIP_ID)
        )';
    ELSE
        RAISE conversation_data_table_exsists;
    END IF;
END;
/

```

```

END IF;
EXCEPTION
    WHEN conversation_data_table_exsists THEN
        dbms_output.put_line('SNDB_conversation_data Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_conversation_data Table already in use by existing
object');
        END IF;
    END;
/

```

```

ALTER TABLE SNDB_conversation_data
ADD CONSTRAINT Fk_frd_user_start_id
    FOREIGN KEY (conversation_initiated_id)
    REFERENCES SNDB_user_account (user_id);

```

```

ALTER TABLE SNDB_conversation_data add CONSTRAINT FK_participant_id FOREIGN KEY
(participant_id)
    REFERENCES SNDB_user_account(user_id);

```

--MESSAGE DATA TABLE CREATION--

```

DECLARE
    p_message_data NUMBER;
    p_message_data_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_message_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_message_data';
    IF p_message_data = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_message_data(
            message_id number NOT NULL,
            messsage_content varchar2(1000),
            message_timestamp timestamp,
            conversation_id number NOT NULL,
            from_message_id number NOT NULL,
            to_message_id number NOT NULL,
            CONSTRAINT message_id_pk PRIMARY KEY (message_id ),

```

```

CONSTRAINT FK_conve_mssg_id FOREIGN KEY (conversation_id )
    REFERENCES SNDB_conversation_data(conversation_PRI_id)

);

ELSE
    RAISE p_message_data_exsists;
END IF;

EXCEPTION
    WHEN p_message_data_exsists THEN
        dbms_output.put_line('SNDB_message_data Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_message_data Table already in use by existing object');
        END IF;
END;
/

```

```

ALTER TABLE SNDB_message_data
ADD CONSTRAINT Fk_From_messg_id FOREIGN KEY (from_message_id)
    REFERENCES SNDB_USER_ACCOUNT(USER_ID);

```

```

ALTER TABLE SNDB_message_data
ADD CONSTRAINT Fk_To_messg_id FOREIGN KEY (to_message_id)
    REFERENCES SNDB_USER_ACCOUNT(USER_ID);

```

```

ALTER TABLE SNDB_MESSAGE_DATA
ADD CONSTRAINT UQ_conversation_id Unique (conversation_id);

```

--Payment Request TABLE CREATION--

```

DECLARE
    p_payment_request_data NUMBER;
    payment_request_data_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_payment_request_data
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_payment_request_data';

```

```

IF p_payment_request_data = 0 THEN
  EXECUTE IMMEDIATE 'CREATE TABLE SNDB_payment_request_data(
    payment_request_id number NOT NULL ,
    requester_id number NOT NULL,
    reciever_id number NOT NULL,
    request_status varchar2(10) NOT NULL,
    request_amount number NOT NULL,
    payment_description varchar2(50) ,
    payment_date date NOT NULL,
    payment_method varchar2(50) NOT NULL,
    conversation_id number NOT NULL,
    CONSTRAINT payment_reqt_id_pk PRIMARY KEY (payment_request_id ) )';

ELSE
  RAISE payment_request_data_table_exsists ;
END IF;
EXCEPTION
  WHEN payment_request_data_table_exsists THEN
    dbms_output.put_line('SNDB_payment_request_data Table already exists');
  WHEN OTHERS THEN
    IF SQLCODE = -955 THEN
      dbms_output.put_line('SNDB_payment_request_data Table already in use by existing
object');
    END IF;
  END;
/

```

```

ALTER TABLE SNDB_payment_request_data
ADD CONSTRAINT Fk_paymnt_reciever_id FOREIGN KEY (reciever_id)
  REFERENCES SNDB_user_account (user_id);

```

```

ALTER TABLE SNDB_payment_request_data
ADD CONSTRAINT Fk_paymnt_requester_id FOREIGN KEY (requester_id)
  REFERENCES SNDB_user_account (user_id);

```

```

ALTER TABLE SNDB_payment_request_data
ADD CONSTRAINT FK_conversation_id FOREIGN KEY (conversation_id)
  REFERENCES SNDB_Message_data(conversation_id);

```

```

--group Table creation--
DECLARE
    p_group NUMBER;
    group_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_group
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_group';

    IF p_group = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_group(
            group_id number NOT NULL,
            group_name varchar2(50) NOT NULL,
            created_date date,
            ls_group_active varchar2(10) NOT NULL,
            CONSTRAINT group_id_pk PRIMARY KEY (group_id)
        )';
    ELSE
        RAISE group_table_exsists ;
    END IF;
EXCEPTION
    WHEN group_table_exsists THEN
        dbms_output.put_line('SNDB_group Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_group Table already in use by existing object');
        END IF;
    END;
/

```

--user_account_group data Table creation--

```

DECLARE
    p_user_account_group NUMBER;
    user_account_group_table_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)

```

```

INTO p_user_account_group
FROM
    user_tables
WHERE
    table_name = 'SNDB_user_account_group';

IF p_user_account_group = 0 THEN
    EXECUTE IMMEDIATE 'CREATE TABLE SNDB_user_account_group(
        user_account_id number NOT NULL,
        group_id number NOT NULL,
        CONSTRAINT FK_user_group_account_id FOREIGN KEY (user_account_id)
            REFERENCES SNDB_user_account(user_id),
        CONSTRAINT FK_user_MSSG_id FOREIGN KEY (group_id)
            REFERENCES SNDB_GROUP(GROUP_id)
    )';

ELSE
    RAISE user_account_group_table_exsists ;
END IF;
EXCEPTION
    WHEN user_account_group_table_exsists THEN
        dbms_output.put_line('SNDB_user_account_group Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_user_account_group Table already in use by existing
object');
        END IF;
END;
/

```

```

ALTER TABLE SNDB_user_account_group
ADD CONSTRAINT user_group_ID PRIMARY KEY (user_account_id, group_id);

```

--GROUP RECIPIENT table creation--

```

DECLARE
    p_group_recipient NUMBER;
    p_group_recipient_exsists EXCEPTION;
BEGIN

```

```

SELECT
    COUNT(*)
INTO p_group_recipient
FROM
    user_tables
WHERE
    table_name = 'SNDB_group_recipient';

IF p_group_recipient = 0 THEN
    EXECUTE IMMEDIATE 'CREATE TABLE SNDB_group_recipient(
        group_recipient_id number NOT NULL,
        group_id number NOT NULL,
        user_id number NOT NULL,
        created_date date NOT NULL,
        is_group_active varchar2(10),
        group_message_id number NOT NULL unique,
        CONSTRAINT group_recipnt_id_pk PRIMARY KEY (group_recipient_id )
    )';
ELSE
    RAISE p_group_recipient_exsists;
END IF;
EXCEPTION
    WHEN p_group_recipient_exsists THEN
        dbms_output.put_line('SNDB_group_recipient Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_group_recipient Table already in use by existing object');
        END IF;
END;
/

```

```

alter table SNDB_group_recipient
add CONSTRAINT FK_user_grp_ID_key FOREIGN KEY (user_id,group_id)
    REFERENCES SNDB_USER_ACCOUNT_GROUP(user_account_id,group_id);

```

--reminder_freq Table creation--

```

DECLARE
    p_reminder_freq NUMBER;
    reminder_freq_exsists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_reminder_freq

```

```

FROM
    user_tables
WHERE
    table_name = 'SNDB_reminder_freq';
IF p_reminder_freq = 0 THEN
    EXECUTE IMMEDIATE 'CREATE TABLE SNDB_reminder_freq(
        reminder_id number NOT NULL,
        freq_type varchar2(20) NOT NULL,
        is_active varchar2(20) NOT NULL,
        CONSTRAINT reminder_id_pk PRIMARY KEY (reminder_id )
    )';
ELSE
    RAISE reminder_freq_exists;
END IF;
EXCEPTION
    WHEN reminder_freq_exists THEN
        dbms_output.put_line('SNDB_reminder_freq Table already exists');
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            dbms_output.put_line('SNDB_reminder_freq Table already in use by existing object');
        END IF;
END;
/
-- group_message Table creation--
DECLARE
    p_group_message NUMBER;
    p_group_message_table_exists EXCEPTION;
BEGIN
    SELECT
        COUNT(*)
    INTO p_group_message
    FROM
        user_tables
    WHERE
        table_name = 'SNDB_group_message';

    IF p_group_message = 0 THEN
        EXECUTE IMMEDIATE 'CREATE TABLE SNDB_group_message(
            group_message_PRI_id number NOT NULL,
            group_message_id number NOT NULL,
            subject varchar2(50),
            creator_id number NOT NULL,
            group_message_content clob NOT NULL,

```

```

group_message_created_date date NOT NULL,
Is_reminder varchar2(10) NOT NULL,
NextReminder_date date NOT NULL,
reminder_freq_id number NOT NULL,
reminder_expiry_date date NOT NULL,
group_message_status varchar2(10),
CONSTRAINT group_message_id_pk PRIMARY KEY (group_message_PRI_id ),
CONSTRAINT FK_creator_id FOREIGN KEY (creator_id)
REFERENCES SNDB_user_account(user_id),
CONSTRAINT FK_RECIP_MESSAGE_id FOREIGN KEY (group_message_id)
REFERENCES SNDB_GROUP_RECIPIENT(group_message_id),
CONSTRAINT FK_reminder_frequ_id FOREIGN KEY (reminder_freq_id)
REFERENCES SNDB_reminder_freq(reminder_id)
)';
ELSE
RAISE p_group_message_table_exsists;
END IF;
EXCEPTION
WHEN p_group_message_table_exsists THEN
dbms_output.put_line('SNDB_group_message Table already exists');
WHEN OTHERS THEN
IF SQLCODE = -955 THEN
dbms_output.put_line('SNDB_group_message Table already in use by existing object');
END IF;
END;

/

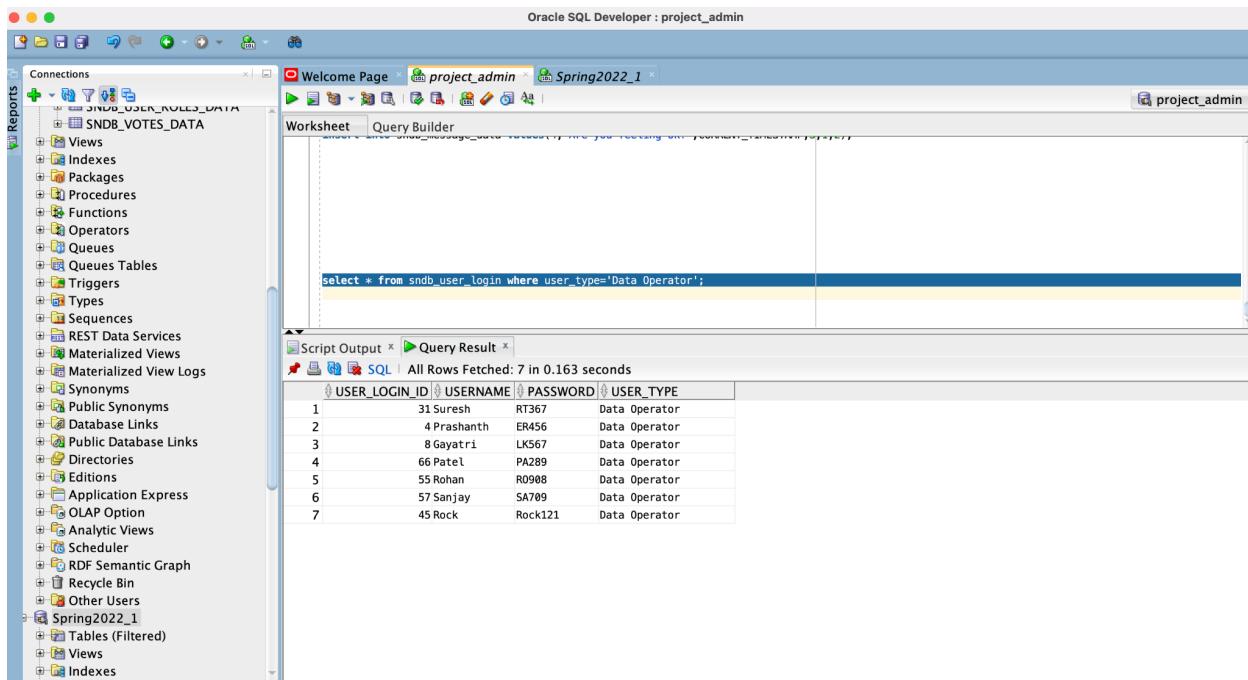
```

Reports:

Scenario I:

The logged user can be user, developer, Data operator, admin, and tester.

Script: select * from sndb_user_login where user_type='Data Operator';



The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree displays a connection named 'project_admin' which includes a schema named 'SNDB_VOTES_DATA'. The main workspace contains a 'Worksheet' tab with the SQL query: 'select * from sndb_user_login where user_type='Data Operator';'. Below the worksheet is a 'Query Result' tab showing the output of the query. The result is a table with the following data:

	USER_LOGIN_ID	USERNAME	PASSWORD	USER_TYPE
1	31 Suresh	RT367	Data Operator	
2	4 Prashanth	ER456	Data Operator	
3	8 Gayatri	LK567	Data Operator	
4	66 Patel	PA289	Data Operator	
5	55 Rohan	R0908	Data Operator	
6	57 Sanjay	SA709	Data Operator	
7	45 Rock	Rock121	Data Operator	

Here, we are checking the user role

```

    select * from sndb_user_login where user_type='User';

```

	USER_LOGIN_ID	USERNAME	PASSWORD	USER_TYPE
1	21	Rajesh	B123	User
2	22	Prajesh	BC123	User
3	23	Vishnu	CD123	User
4	25	Sharan	FE456	User
5	24	Geetha	DE345	User
6	26	Sasank	GF234	User
7	27	Srujan	KJ245	User
8	28	Pranay	KI567	User
9	29	Prasad	KI453	User
10	30	Ramesh	L0987	User
11	32	Srinivas	BH765	User
12	33	Aruna	IJ345	User
13	34	Pramod	KJ345	User
14	35	Harsha	NH789	User
15	36	Akhil	VB234	User
16	37	Shubam	VG473	User
17	38	Chaitanya	CH456	User

To get user_id, first_name and last_name where dob of an user is less than 29-Mar-99

Script: select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login ,
sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.dob < '29-Mar-99';

Oracle SQL Developer File Edit View Navigate Run Source Team Tools Window Help Fri Apr 15 1:40 PM Oracle SQL Developer : project_admin

Connections SNDB_USER_ROLES_DATA

Views Indexes Packages Procedures Functions Operators Queues Queues Tables Triggers Types Sequences REST Data Services Materialized Views Materialized View Logs Synonyms Public Synonyms Database Links Public Database Links Directories Editions Application Express OLAP Option Analytic Views Scheduler RDF Semantic Graph Recycle Bin Other Users Spring2022_1 Tables (Filtered) Views Indexes

Worksheet Query Builder

```
select * from sndb_user_login where user_type='User';
select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.dob < '29-Mar-99';
```

Script Output Query Result All Rows Fetched: 7 in 0.184 seconds

USER_LOGIN_ID	FIRST_NAME	LAST_NAME
1	3 Ben	lee
2	4 John	Baig
3	5 Rug	Men
4	7 Rock	Sack
5	9 Andrew	Sin
6	10 Mandy	Sen
7	12 Andrew	Ping

Messages - Log Click on an identifier with the Command key down to perform "Go to Declaration" Line 162 Column 73 Insert Modified Unix/Mac: LF

The users who joined the application in April

Script: select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.dob < '29-Apr-99';

Oracle SQL Developer File Edit View Navigate Run Source Team Tools Window Help Fri Apr 15 1:43 PM Oracle SQL Developer : project_admin

Connections SNDB_USER_ROLES_DATA

Views Indexes Packages Procedures Functions Operators Queues Queues Tables Triggers Types Sequences REST Data Services Materialized Views Materialized View Logs Synonyms Public Synonyms Database Links Public Database Links Directories Editions Application Express OLAP Option Analytic Views Scheduler RDF Semantic Graph Recycle Bin Other Users Spring2022_1 Tables (Filtered) Views Indexes

Welcome Page project_admin Spring2022_1

Worksheet Query Builder

```
select * from sndb_user_login where user_types='User';
select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.dob < '29-Mar-99';
select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.created_at > '14-APR-22';
```

Script Output SQL All Rows Fetched: 12 in 0.192 seconds

USER_LOGIN_ID	FIRST_NAME	LAST_NAME
1	Mee	Fnu
2	Sud	lnu
3	Ben	lee
4	John	Baig
5	Rug	Men
6	Sally	And
7	Rock	Sack
8	Caitlyn	Brandy
9	Andrew	Sin
10	Mandy	Sen
11	Cat	Man
12	Andrew	Ping

Messages - Log Click on an identifier with the Command key down to perform "Go to Declaration" Line 165 Column 71 Insert Modified Unix/Mac: LF

Count of users who joined the application after Jan-97

Script: select count(*) from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.created_at > '14-Jan-97';

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar on the left lists several database objects under 'SNDDB_USER_ROLES_DATA'. The 'Worksheet' tab in the center contains a SQL query:

```
select * from sndb_user_login where user_types='User';
select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.dob < '29-Mar-99';
select count(*) from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.created_at > '14-Jan-97';
```

The 'Script Output' tab shows the result of the query:

COUNT(*)
12

The status bar at the bottom indicates 'Line 165 Column 59 | Insert | Modified! Unix/Mac: LF'.

Users who are elder than 18 years

Script: select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login ,
sndb_user_account acc
where login.user_login_id= acc.user_account_id and months_between(created_at,dob)/12 >18;

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays a tree view of database objects under the connection 'SNDB_USER_ROLES_DATA'. The 'Tables' node is expanded, showing 'SNDB_VOTES_DATA'. The 'Worksheet' tab in the center contains a SQL query:

```

select * from sndb_user_login where user_type='User';

select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.dob < '29-Mar-99';

select count(*) from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.created_at > '14-Jan-97';

select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and months_between(created_at,dob)/12 >18;

```

The 'Query Result' tab below shows the output of the last query, which lists 10 rows of user information:

USER_LOGIN_ID	FIRST_NAME	LAST_NAME
1	3Ben	lee
2	4John	Baig
3	5Rug	Men
4	6Sally	And
5	7Rock	Sack
6	8Caitlyn	Brandy
7	9Andrew	Sin
8	10Mandy	Sen
9	11Cat	Man
10	12Andrew	Ping

To get the count of users who joined for the last three years month wise

```

select count(*),TO_CHAR (created_at, 'MON')|| ' '|| TO_CHAR (created_at, 'YYYY') month_year
from sndb_user_account
where TRUNC (MONTHS_BETWEEN (SYSDATE, created_at)) < 36
group by TO_CHAR (created_at, 'MON')|| ' '|| TO_CHAR (created_at, 'YYYY');

```

COUNT(*)	MONTH_YEAR
1	1 APR 2021
2	10 FEB 2022
3	30 APR 2022
4	9 JAN 2022
5	20 MAR 2021

 The status bar at the bottom shows 'Line 171 Column 1' and 'Modified Unix/Mac: LF'. The Mac OS X dock is visible at the very bottom."/>

```

select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and months_between(created_at,dob)/12 >18;

select count(*),TO_CHAR (created_at, 'MON')|| ' '|| TO_CHAR (created_at, 'YYYY') month_year
from sndb_user_account
where TRUNC (MONTHS_BETWEEN (SYSDATE, created_at)) < 36
group by TO_CHAR (created_at, 'MON')|| ' '|| TO_CHAR (created_at, 'YYYY');
    
```

COUNT(*)	MONTH_YEAR
1	1 APR 2021
2	10 FEB 2022
3	30 APR 2022
4	9 JAN 2022
5	20 MAR 2021

To get the records of users whose friendship request is pending

Script: select a.friendship_id,b.status_type

from sndb_addfriend_data a,sndb_status b

where a.status_id=b.status_id and b.status_type='Pending';

```

select * from sndb_addfriend_data;
select * from sndb_status;
select * from sndb_user_account;

select a.friendship_id,b.status_type,c.first_name as requester_name,c.first_name as addressee_name
from sndb_addfriend_data a,sndb_status b, sndb_user_account c
where a.status_id=b.status_id and a.requester_id = c.user_id;

select a.friendship_id,b.status_type
from sndb_addfriend_data a,sndb_status b
where a.status_id=b.status_id and b.status_type='Pending';

```

FRIENDSHIP_ID	STATUS_TYPE
1	3 Pending
2	26 Pending
3	7 Pending
4	8 Pending
5	13 Pending
6	17 Pending

To get the requester name, addressee name and request status

Script: select a.friendship_id,b.status_type,(select c.first_name from sndb_user_account c where a.requester_id = c.user_id) as Requester_Name,(select c.first_name from sndb_user_account c where a.addressee_id = c.user_id)as addressee_name from sndb_addfriend_data a,sndb_status b where a.status_id=b.status_id;

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays various database objects under the 'Connections' section, including 'DB-unimodel', 'project_admin', 'Spring2022_1', 'Spring2022_2', and 'tester1'. The 'tester1' connection is expanded, showing 'Tables (Filtered)', 'Views', 'Indexes', 'Packages', 'Procedures', 'Functions', 'Operators', 'Queues', 'Queues Tables', 'Triggers', 'Types', 'Sequences', 'REST Data Services', 'Materialized Views', 'Materialized View Logs', 'Synonyms', 'Public Synonyms', 'Database Links', 'Public Database Links', 'Directories', 'Editions', 'Application Express', 'OLAP Option', 'Analytic Views', 'Scheduler', and 'RDF Semantic Graph'. The 'tester1' connection is also selected in the top bar.

The main area consists of two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab contains the following SQL code:

```

select user_id, count(post_id) from sndb_post_data
group by user_id;

select post_id, upcount, downcount from sndb_votes_data;

select a.friendship_id, b.status_type, (select c.first_name from sndb_user_account c
where a.requester_id = c.user_id) as Requester_Name, (select c.first_name from sndb_user_account c
where a.addressee_id = c.user_id) as Addressee_Name
from sndb_addfriend_data a, sndb_status b
where a.status_id = b.status_id;

```

The 'Query Result' tab shows the output of the last query, which fetched 50 rows in 0.122 seconds. The results are presented in a table:

FRIENDSHIP_ID	STATUS_TYPE	REQUESTER_NAME	ADDRESSE_NAME
1	2Rejected	John	Rug
2	3Pending	Rug	Ben
3	4Accepted	John	Sally
4	31Accepted	Harsha	Pramod
5	32Accepted	Akhil	Aruna
6	33Accepted	Shubam	Srinivas
7	34Accepted	Aditya	Suresh
8	35Rejected	Sunny	Ramesh
9	36Accepted	Krishna	Prasad
10	37Rejected	John	Pranay
11	38Accepted	Ben	Srujan
12	39Accepted	Rug	Sasank
13	40Accepted	Sally	Sharan
14	61Accepted	Subhash	Andrew
15	62Rejected	Rock	Sharan
16	63Rejected	Caitlyn	Sasank
17	EXCLUDED	Andrea	Emilia

Count of pending, accepted and rejected requests

Script: select a.status_type, count(*)
 from sndb_addfriend_data b, sndb_status a
 where b.status_id = a.status_id
 group by status_type;

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The title bar indicates "Oracle SQL Developer : project_admin". The left sidebar displays a tree view of database objects under "DB-unimodel" and "project_admin", including Tables, Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Triggers, Types, Sequences, REST Data Services, Materialized Views, Materialized View Logs, Synonyms, Public Synonyms, Database Links, Public Database Links, Directories, Editions, Application Express, OLAP Option, Analytic Views, Scheduler, and RDF Semantic Graph. The main workspace contains a "Worksheet" tab and a "Query Builder" tab. The "Query Builder" tab contains the following SQL code:

```

select a.friendship_id,b.status_type,c.first_name as requester_name,c.first_name as addressee_name
from sndb_addfriend_data a,sndb_status b, sndb_user_account c
where a.status_id=b.status_id and a.requester_id = c.user_id;

select a.friendship_id,b.status_type
from sndb_addfriend_data a,sndb_status b
where a.status_id=b.status_id and b.status_type='Pending';

select a.status_type, count(*)
from sndb_addfriend_data b, sndb_status a
where b.status_id = a.status_id
group by status_type;

```

The "Script Output" window below shows the results of the third query:

STATUS_TYPE	COUNT(*)
1 Rejected	21
2 Pending	6
3 Accepted	43

The bottom status bar shows "Messages - Log", "Click on an identifier with the Command key down to perform "Go to Declaration"" and "Line 192 Column 33 | Insert | Modified! Unix/Mac LF".

Scenario 2:

Number of posts by each user

Script: select user_id,count(post_id) from sndb_post_data
group by user_id;

Oracle SQL Developer : project_admin

Connections

- Oracle Connections
- DB-unimodel
- project_admin
- Spring2022_1
- Spring2022_2
- tester1
- Tables (Filtered)
- Views
- Indexes
- Packages
- Procedures
- Functions
- Operators
- Queues
- Queues Tables
- Triggers
- Types
- Sequences
- REST Data Services
- Materialized Views
- Materialized View Logs
- Synonyms
- Public Synonyms
- Database Links
- Public Database Links
- Directories
- Editions
- Application Express
- OLAP Option
- Analytic Views
- Scheduler
- RDF Semantic Graph

Worksheet Query Builder

```

select * from sndb_post_data;
select * from sndb_votes_data;

select * from sndb_post_data a, sndb_user_account b
where a.user_id = b.user_id and a.user_id=21;

insert into sndb_post_data values (103, 'Just trying', 'one person many posts',23);

select user_id,count(post_id) from sndb_post_data
group by user_id;

```

Script Output Fetched 50 rows in 0.111 seconds

USER_ID	COUNT(POST_ID)
1	23
2	27
3	6
4	14
5	51
6	52
7	57
8	58
9	64
10	50
11	31
12	36
13	40
14	7
15	15
16	59
17	1

Likes and dislikes on each post

Script: select post_id,upcount,downcount from sndb_votes_data;

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays database connections and objects. The main area has three tabs: 'Worksheet' (Query Builder), 'Script Output', and 'Query Result'. The 'Worksheet' tab contains a SQL script with several select statements and an insert statement. The 'Query Result' tab shows the output of the last select statement, which is a table with four columns: POST_ID, UPCOUNT, and DOWNCOUNT. The data is as follows:

POST_ID	UPCOUNT	DOWNCOUNT
1	1	2
2	3	4
3	4	5
4	5	3
5	6	13
6	9	4
7	10	3
8	11	3
9	7	14
10	22	20
11	23	4
12	24	3
13	25	2
14	26	1
15	27	7
16	28	4
17	29	1

SCENARIO 3:

Number of conversations started in each week

Script:

```
Select TRUNC(cast(conversation_started as date),'IW') as "Weekly(Week start date)", count(*)
as Number_of_conversations from sndb_conversation_data group by
TRUNC(cast(conversation_started as date),'IW');
```

The screenshot shows the Oracle SQL Developer interface. In the top panel, there is a 'Worksheet' tab with a query builder containing the following SQL code:

```
Select TRUNC(cast(conversation_started as date), 'IW') as "Weekly(Week start date)", count(*) as Number_of_conversations from sndb_conversation_data group by TRUNC(cast(conversation_started as date), 'IW');
```

Below the worksheet is a 'Query Result' tab showing the execution of the query. The results are displayed in a table:

Weekly(Week start date)	NUMBER_OF_CONVERSATIONS
1 21-MAR-22	3
2 11-APR-22	34
3 07-MAR-22	11
4 28-MAR-22	6
5 14-MAR-22	6
6 04-APR-22	5
7 28-FEB-22	6

The results indicate the number of conversations started each week, with the highest volume occurring in week 2 (11-APR-22) at 34 conversations.

Conversations Started by first and last name and their content

Script:

```
select ua.first_name,ua.last_name,md.messsage_content from sndb_message_data md inner  
join sndb_conversation_data cd on cd.conversation_pri_id=md.conversation_id  
inner join sndb_user_account ua on ua.user_id=cd.conversation_initiated_id;
```

The screenshot shows the MySQL Workbench interface. In the 'Query Builder' tab, a SQL query is displayed:

```
select ua.first_name,ua.last_name,md.message_content from sndb_message_data md inner join sndb_conversation_data cd on cd.conversation_initiated_id; inner join sndb_user_account ua on ua.user_id=cd.conversation_initiated_id;
```

In the 'Query Result' tab, the output is shown in a table format:

	FIRST_NAME	LAST_NAME	MESSAGE_CONTENT
1	John	Baig	Whats up
2	Rug	Men	Glad to meet you
3	Rug	Men	We are here
4	Swapnil	Danailla	Will do it
5	Danish	Sait	Glad to join
6	Charan	Tilak	Nice to see you
7	Prabhas	Dilwar	Give me food
8	Govind	Bhawas	Whats up
9	Patel	Singh	Eat food
10	Bagath	Vilash	We are here
11	Sardar	Mohanlal	Glad to join
12	Subhash	Banwarlal	Thank you for support
13	Swapnil	Danailla	Will do it
14	Danish	Sait	Glad to join
15	Rawath	Kumar	Thank you for support
16	Modi	Narendra	Whats up
17	Pramod	Datta	Glad to join
18	Trupti	Kamat	Thank you for support
19	Rohan	Kohli	Nice to see you
20	Sushruth	Beeti	Give me food
21	Saniav	Chandnani	Eat food

Number of Payment Requests Approved/Denied

Script:

```
select request_status as Payment_request,count(*) as count from sndb_payment_request_data
group by request_status;
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a toolbar with various icons. The main area is divided into two sections: 'Worksheet' and 'Query Builder'. The 'Worksheet' section contains a SQL query:`select request_status as Payment_request, count(*) as count from sndb_payment_request_data group by request_status;`In the bottom-right pane, titled 'Query Result', the output is displayed in a table:| PAYMENT_REQUEST | COUNT |
| --- | --- |
| 1 Denied | 7 |
| 2 Approved | 32 |

The message 'All Rows Fetched: 2 in 0.116 seconds' is shown above the table.

SCENARIO 4

Number of groups created in the week

Script:

```
select TRUNC(CREATED_DATE,'IW') as "Weekly(Week Start Date)",count(*) as "Number of groups created" from sndb_group group by TRUNC(CREATED_DATE,'IW');
```

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page', 'project_admin', and 'project_admin~1'. The main area is titled 'Worksheet' and contains a 'Query Builder' window with the following SQL code:

```
select TRUNC(CREATED_DATE,'IW') as "Weekly(Week Start Date)",count(*) as "Number of groups created" from sndb_group group by TRUNC(CREATED_DATE,'IW');
```

Below the query builder is a 'Query Result' window showing the output:

Weekly(Week Start Date)	Number of groups created
1 11-APR-22	2
2 07-FEB-22	2
3 07-MAR-22	3
4 28-MAR-22	2
5 04-APR-22	6

Number of active and inactive groups

Script:

```
select IS_GROUP_ACTIVE as "GROUP Active (Yes/NO)",count(*) as "Count" from sndb_group  
group by IS_GROUP_ACTIVE;
```

The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page', 'project_admin', and 'project_admin~1'. The main area is titled 'Worksheet' and contains a 'Query Builder' window with the following SQL code:

```
select IS_GROUP_ACTIVE as "GROUP Active (Yes/NO)",count(*) as "Count" from sndb_group group by IS_GROUP_ACTIVE;
```

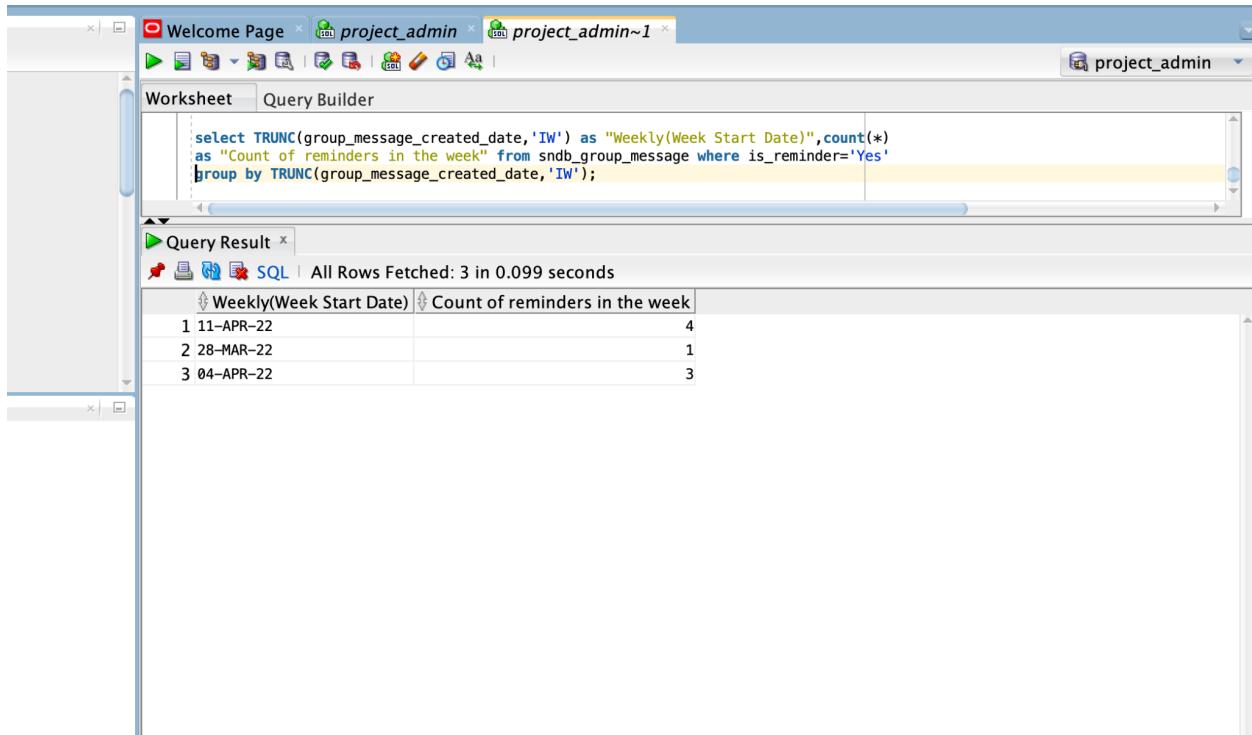
Below the query builder is a 'Query Result' window showing the output:

GROUP Active (Yes/NO)	Count
No	1
Yes	14

Number of reminders created in a week

Script:

```
select TRUNC(group_message_created_date,'IW') as "Weekly(Week Start Date)",count(*)  
as "Count of reminders in the week" from sndb_group_message where isReminder='Yes'  
group by TRUNC(group_message_created_date,'IW');
```



The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the executed SQL query. The 'Query Result' tab shows the output, which is a table with three rows. The columns are 'Weekly(Week Start Date)' and 'Count of reminders in the week'. The data is as follows:

Weekly(Week Start Date)	Count of reminders in the week
1 11-APR-22	4
2 28-MAR-22	1
3 04-APR-22	3

Insert statement:

```
insert into sndb_user_roles_data values (6,'Admin','Testing access');  
insert into sndb_user_roles_data values (3,'Tester','Testing access');  
insert into sndb_user_roles_data values (4,'Data Operator','Data transact access');  
insert into sndb_user_roles_data values (5,'Developer','Hybrid access');
```

```
insert into sndb_user_login values(3,'Sagar','QW123','Admin' );  
insert into sndb_user_login values(4,'Prashanth','ER456','Data Operator');  
insert into sndb_user_login values(5,'Vijay','TY789','Admin');
```

```

insert into sndb_gender_data values(1,'FEMALE');
insert into sndb_gender_data values(2,'MALE');
insert into sndb_gender_data values(3,'OTHERS');

INSERT INTO SNDB_USER_ACCOUNT
VALUES(100,3,'Meenu','Ahale','Ahale.m@gmail.com','6789011234','Massachusetts Institute of
Technology',
'College of Engineering','Network Structures',1,TO_DATE('12-NOV-1994',
'DD-MON-YY'),400,TO_TIMESTAMP('30-APR-22 09:02:44', 'DD-MON-YY HH24:mi:ss'));
INSERT INTO SNDB_USER_ACCOUNT
VALUES(101,4,'Sudha','Aadekar','Aadekar.s@gmail.com','6789015678','University of New
Hampshire','Khoury College of CS',' Cloud Computing',
1,TO_DATE('12-JUN-1992', 'DD-MON-YY'),401,TO_TIMESTAMP('01-APR-22 23:30:44',
'DD-MON-YY HH24:mi:ss'));
INSERT INTO SNDB_USER_ACCOUNT
VALUES(102,5,'Sagar','Ambedkar','Ambedkar.s@gmail.com','6788029012','Boston
University','College of Science',
'Operating Systems',2,TO_DATE('21-JUL-1982',
'DD-MON-YY'),402,TO_TIMESTAMP('03-APR-22 12:08:54', 'DD-MON-YY HH24:mi:ss'));

insert into sndb_post_data VALUES(1,'Heyy','Whats up',3);
insert into sndb_post_data VALUES(2,'Heyy','Whats up',4);
insert into sndb_post_data VALUES(3,'Heyy','Eat it',5);

insert into sndb_votes_data VALUES(1,1,4,23);
insert into sndb_votes_data VALUES(2,2,10,18);
insert into sndb_votes_data VALUES(3,3,9,21);

insert into sndb_logged_in_data values(1,3,TO_TIMESTAMP('26-FEB-22
20.11.12','DD-MON-YY HH24:mi:ss'),TO_TIMESTAMP('26-FEB-22 16.11.58', 'DD-MON-YY
HH24:mi:ss'));
insert into sndb_logged_in_data values(2,4,TO_TIMESTAMP('26-FEB-22
20.11.12','DD-MON-YY HH24:mi:ss'),TO_TIMESTAMP('26-FEB-22 16.11.58', 'DD-MON-YY
HH24:mi:ss'));
insert into sndb_logged_in_data values(3,5,TO_TIMESTAMP('26-FEB-22
20.11.12','DD-MON-YY HH24:mi:ss'),TO_TIMESTAMP('26-FEB-22 16.11.58', 'DD-MON-YY
HH24:mi:ss'));

insert into sndb_user_photo_data values(1,3,'http://hrtf//me',TO_TIMESTAMP('03-MAR-22
05.11.42', 'DD-MON-YY HH24:mi:ss'),'Yes');
insert into sndb_user_photo_data values(2,4,'http://tinyurl/me',TO_TIMESTAMP('05-MAR-22
02.07.54', 'DD-MON-YY HH24:mi:ss'),'Yes');

```

```
insert into sndb_user_photo_data values(3,5,'http://tinyurl/you',TO_TIMESTAMP('06-MAR-22  
05.09.59', 'DD-MON-YY HH24:mi:ss'),'No');
```

```
insert into sndb_status values (1, 'Pending');  
insert into sndb_status values (2, 'Rejected');  
insert into sndb_status values (3, 'Accepted');
```

```
insert into sndb_addfriend_data  
values(1,3,4,CURRENT_TIMESTAMP,2,CURRENT_TIMESTAMP,'Great to  
connect',CURRENT_TIMESTAMP,CURRENT_TIMESTAMP);  
insert into sndb_addfriend_data  
values(2,4,5,CURRENT_TIMESTAMP,2,CURRENT_TIMESTAMP,'Great to  
connect',CURRENT_TIMESTAMP,CURRENT_TIMESTAMP);
```

```
insert into sndb_conversation_data  
values(1,1,3,CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,4);  
insert into sndb_conversation_data  
values(2,2,4,CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,5);
```

```
insert into sndb_message_data values(1,'hello, come soon',CURRENT_TIMESTAMP,1,3,4);  
insert into sndb_message_data values(2,'Great Well Soon',CURRENT_TIMESTAMP,2,4,5);
```

```
insert into sndb_payment_request_data values(2000, 3 ,4 , 'Approved' ,1 , 'Shop and  
stop bill', TO_TIMESTAMP('01-MAR-22 04:12.49', 'DD-MON-YY HH24:mi:ss'), 'Debit', 1);  
insert into sndb_payment_request_data values(2001, 4 ,5, 'Denied', 10.05, 'Wallgreens  
bill',TO_TIMESTAMP('04-MAR-22 03.11.50', 'DD-MON-YY HH24:mi:ss'), 'Credit', 2);
```

```
insert into sndb_group values(1,'DBMS',sysdate,'Yes');  
insert into sndb_group values(2,'Team5',sysdate,'No');
```

```
insert into sndb_user_account_group values(3,1);  
insert into sndb_user_account_group values(4,2);
```

```
insert into sndb_group_recipient values(3,1,3,sysdate,'Yes',1);  
insert into sndb_group_recipient values(4,2,4,sysdate,'No',2);
```

```
insert into sndb_reminder_freq values(1,'Weekly','Yes');  
insert into sndb_reminder_freq values(2,'Weekly','No');
```

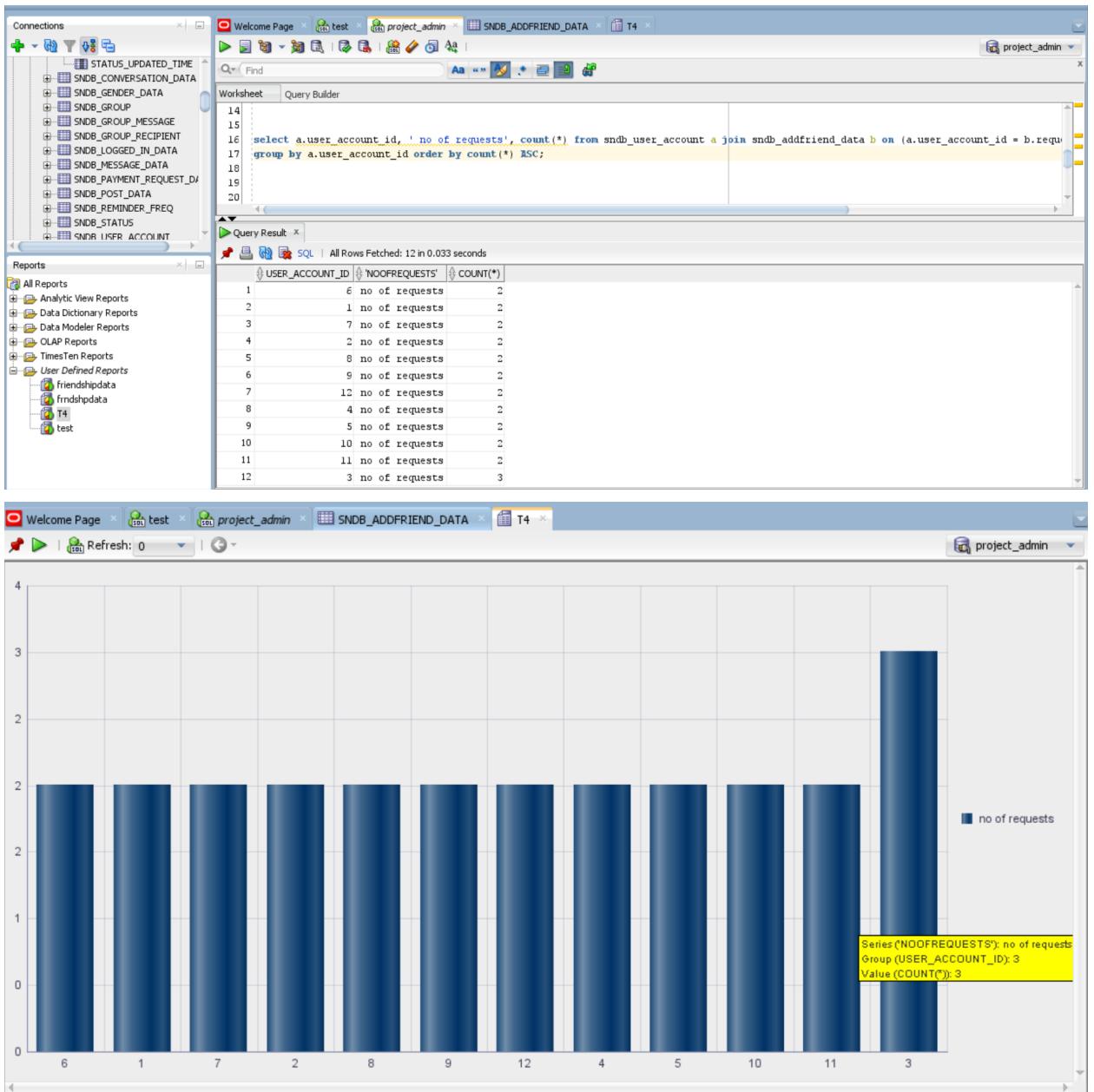
```
insert into sndb_group_message values(3,1,'Hola',3,'How are you  
doing',sysdate,'Yes',sysdate+1,1,sysdate+10,'Sent');  
insert into sndb_group_message values(4,2,'Give me peace',4,'Hope you are doing  
fine',sysdate,'Yes',sysdate+1,2,sysdate+10,'Sent');
```

PROJECT 5 (Team 5)

REPORTS:

1. NO OF FRIEND REQUESTS:

```
select a.user_account_id, 'no of requests', count(*) from sndb_user_account a join  
sndb_addfriend_data b on (a.user_account_id = b.requester_id or a.user_account_id =  
b.addressee_id)  
group by a.user_account_id order by count(*) ASC;
```



2. NO OF PAYMENT REQUESTS:

```

select a.user_account_id, 'no of payment reqs', count(*) from sndb_user_account a join
sndb_payment_request_data b on (a.user_account_id = b.requester_id or a.user_account_id =
b.receiver_id)
group by a.user_account_id order by count(*) ASC;

```

Connections

- SNDB_GROUP_RECIPIENT
- SNDB_LOGGED_IN_DATA
- SNDB_MESSAGE_DATA
- SNDB_PAYMENT_REQUEST_DATA
- PAYMENT_REQUEST_ID
- REQUESTER_ID
- RECEIVER_ID
- REQUEST_STATUS
- REQUEST_AMOUNT
- PAYMENT_DESCRIPTION
- PAYMENT_DATE
- PAYMENT_METHOD
- CONVERSATION_ID

Reports

- All Reports
- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports
- User Defined Reports
- payreq
- reqrec
- rr
- T4
- t5
- t7
- test
- Totalreq

Worksheet

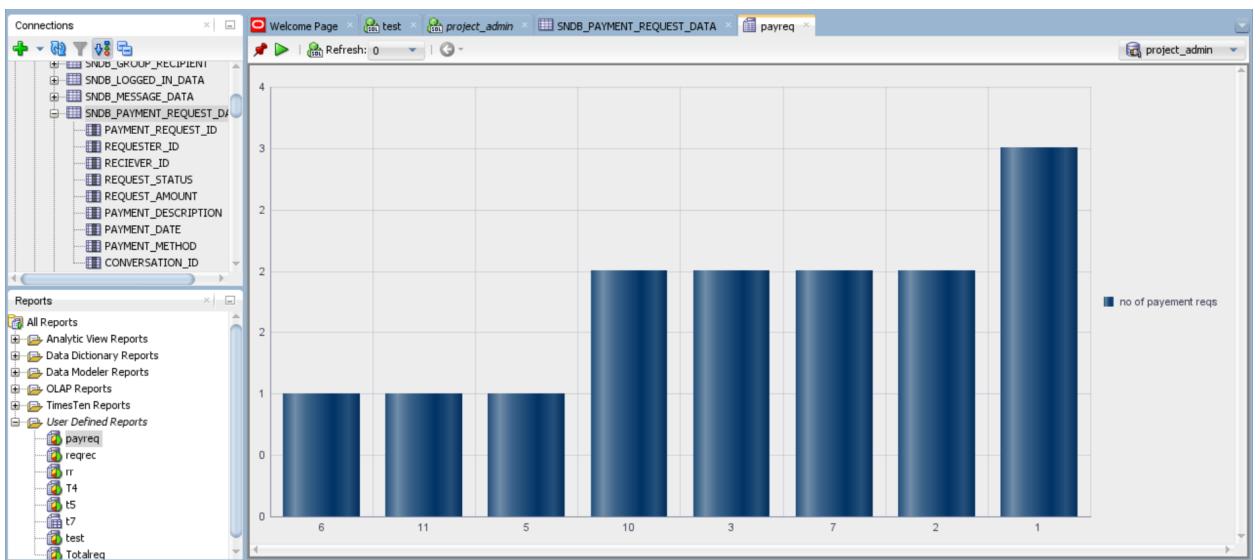
```

39
40
41
42 select a.user_account_id, 'no of payment reqs', count(*) from sndb_user_account a join sndb_payment_request_data b on (a.user_account_id = b.user_account_id)
43 group by a.user_account_id order by count(*) DESC;
44
45
46
47

```

Query Result

USER_ACCOUNT_ID	'NOOFPAYMENTREQS'	COUNT(*)
1	6 no of payment reqs	1
2	11 no of payment reqs	1
3	5 no of payment reqs	1
4	10 no of payment reqs	2
5	3 no of payment reqs	2
6	7 no of payment reqs	2
7	2 no of payment reqs	2
8	1 no of payment reqs	3



3. More than 5 friend requests in a day:

```
select created_datetime,count(*) from sndb_addfriend_data group by created_datetime having count(*)>5
```

The screenshot shows the Oracle SQL Developer interface. In the 'Worksheet' tab, a query is written to find friend requests in a day:

```
/*More than 5 friend requests in a day */
select created_datetime,count(*) from sndb_addfriend_data group by created_datetime having count(*)>5
```

In the 'Query Result' tab, the output is displayed as a table:

CREATED_DATETIME	COUNT(*)
1 15-MAR-22 12.11.54.000000000 PM	10
2 01-MAR-22 04.12.49.000000000 AM	10
3 26-MAR-22 04.11.58.000000000 PM	9
4 20-MAR-22 06.11.56.000000000 PM	9
5 04-MAR-22 03.11.50.000000000 AM	10
6 09-MAR-22 10.08.52.000000000 AM	10
7 27-MAR-22 07.11.57.000000000 PM	9
8 11-MAR-22 02.44.53.000000000 AM	10
9 06-MAR-22 07.22.51.000000000 AM	10
10 17-MAR-22 02.11.55.000000000 AM	10

4. People who do not have letter 'A' in their first name and last name

```
select ua.first_name,ua.last_name,md.message_content from sndb_message_data md inner
join sndb_conversation_data cd on cd.conversation_pri_id=md.conversation_id
inner join sndb_user_account ua on ua.user_id=cd.conversation_initiated_id
where ua.first_name not like '%A%'
and ua.last_name not like '%A%';
```

```

/* People who do not have letter 'A' in their first name and last name*/
select ua.first_name,ua.last_name,md.message_content from sndb_message_data md
inner join sndb_user_account ua on ua.user_id=cd.conversation_initiated_id
where ua.first_name not like '%A%'
and ua.last_name not like '%A%';

```

Query Result

FIRST_NAME	LAST_NAME	MESSAGE_CONTENT
1 VINCENTS	MERILL	We are here
2 LEZLEY	WIGIN	Whats up?
3 CLYVE	ORRITT	Glad to meet you
4 KENDELL	FEIFER	Whats up
5 PENNI	JIROUSEK	Whats up
6 CORY	CUTTLES	Glad to meet you
7 HERCULIE	REGI	Eat food
8 FEBJORIE	BREIT	Glad to join
9 STU	FIENNES	Nice to see you
10 TEDDIE	DESSON	Glad to join
11 INGE	MOORWOOD	Glad to meet you
12 CRISSIE	WOLSEY	We are here
13 LORY	GISBOURN	Thank you for support
14 TESSY	KETTLESTRING	Glad to meet you
15 HYMIE	BRUMBYE	We are here
16 ENOCH	JESSEL	Whats up

5. Number of MALE, FEMALE and Others genders in each university

```

Select * FROM
(
select university_name, gd.name as gender,count(*) as cnt from sndb_user_account ua inner join
sndb_gender_data gd on ua.gender_id=gd.gender_id
group by university_name, gd.name
) t
PIVOT
(
sum(cnt)
for gender IN
('MALE','FEMALE','OTHERS')
) PivotTable;

```

Worksheet Query Builder

```
/* Number of MALE, FEMALE and Others genders in each university*/
Select * FROM
(
  select university_name, gd.name as gender, count(*) as cnt from sndb_user_account ua inner join sndb_gender_data gd on ua.gender_id=gd.gen
  group by university_name, gd.name
) t
PIVOT
(
  sum(cnt)
  for gender IN
  ('MALE','FEMALE','OTHERS')
) PivotTable;
```

Query Result

All Rows Fetched: 9 in 0.132 seconds

UNIVERSITY_NAME	'MALE'	'FEMALE'	'OTHERS'
1 UNIVERSITY OF RHODE ISLAND	11	25	15
2 YALE UNIVERSITY	12	11	3
3 UNIVERSITY OF NEW HAMPSHIRE	28	27	5
4 UNIVERSITY OF NEW HAVEN	5	4	4
5 BOSTON UNIVERSITY	16	17	1
6 MASSACHUSETTS INSTITUTE OF TECHNOLOGY	25	24	14
7 WENTWORTH INSTITUTE OF TECHNOLOGY	4	6	2
8 FAIRFIELD UNIVERSITY	8	4	1
9 UNIVERSITY OF MASSACHUSETTS LOWELL	11	9	6

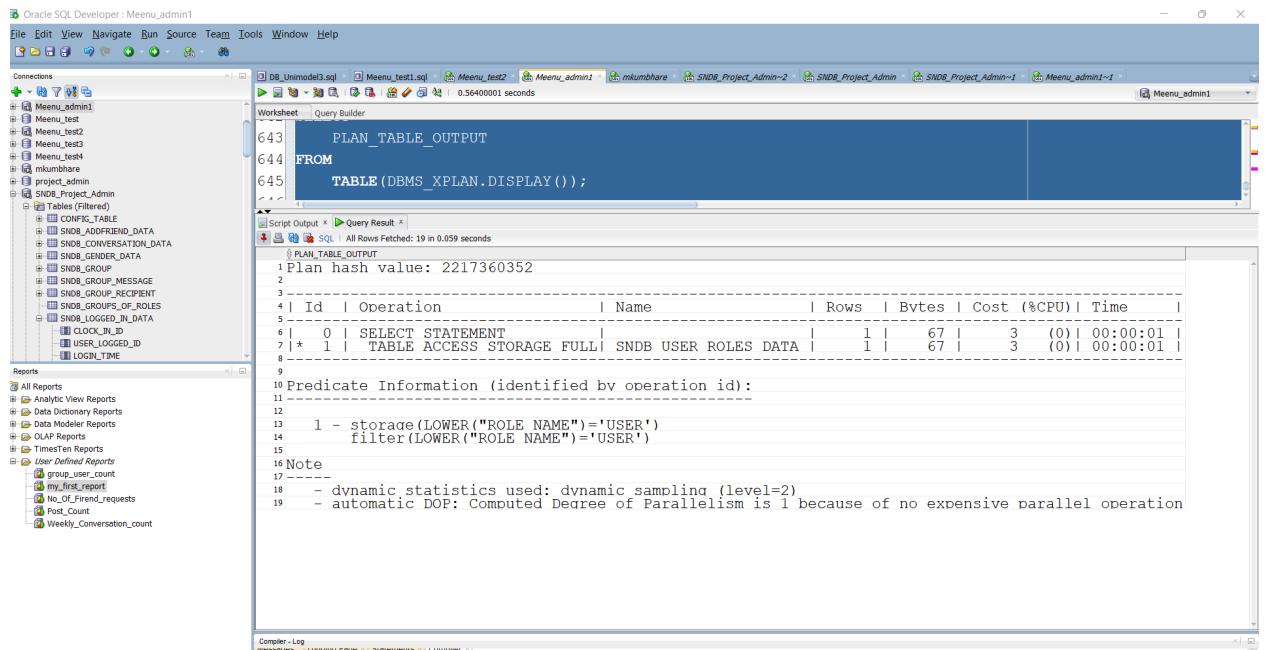
Dbms Output

--Without INDEXES

EXPLAIN PLAN FOR

```
SELECT * FROM sndb_user_roles_data
WHERE lower(Role_name) = 'USER';
```

```
SELECT
  PLAN_TABLE_OUTPUT
FROM
  TABLE(DBMS_XPLAN.DISPLAY());
```



--With function based Index

```
CREATE INDEX SNDB_INDEX_Role_name
ON sndb_user_roles_data(lower(Role_name));
```

EXPLAIN PLAN FOR

```
SELECT * FROM sndb_user_roles_data
WHERE lower(Role_name) = 'USER';
```

```
SELECT
  PLAN_TABLE_OUTPUT
FROM
  TABLE(DBMS_XPLAN.DISPLAY());
drop index SNDB_INDEX_Role_name;
```

Oracle SQL Developer - Meenu_admin1

File Edit View Navigate Run Source Team Tools Window Help

Connections

- Meenu_admin1
- Meenu_test
- Meenu_test2
- Meenu_test3
- Meenu_test4
- mkumbhare
- project_admin
- SNDB_Project_Admin

DB_Unimodel3.sql Meenu_test1.sql Meenu_test2 Meenu_admin1 mkumbhare SNDB_Project_Admin-2 SNDB_Project_Admin-1 Meenu_admin1-1 Meenu_admin1

0.597 seconds

Worksheet Query Builder

```
650 ON sndb_user_roles_data(lower(Role_name));
651 --drop index SNDB INDEX Role name;
```

Script Output | Query Result x

All Rows Fetched: 19 in 0.066 seconds

PLAN_TABLE_OUTPUT
1 Plan hash value: 3012962127
2
3
4 Id Operation Name Rows Bytes Cost (%CPU) Time
5 0 SELECT STATEMENT SNDB USER ROLES DATA 1 94 1 (0) 00:00:0
6 1 TABLE ACCESS BY INDEX ROWID BATCHED SNDB USER ROLES DATA 1 94 1 (0) 00:00:0
7 * 2 INDEX RANGE SCAN SNDB INDEX ROLE NAME 1 94 1 (0) 00:00:0
8
9
10 Predicate Information (identified by operation id):
11 -----
12
13
14 2 - access(LOWER("ROLE NAME")='USER')
15
16 Note
17 -----
18 - dynamic statistics used: dynamic sampling (level=2)
19 - automatic DOP: Computed Degree of Parallelism is 1 because of no expensive parallel operation

Compiler - Log

Messages | Logging Page | Statement | Compiler |