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.

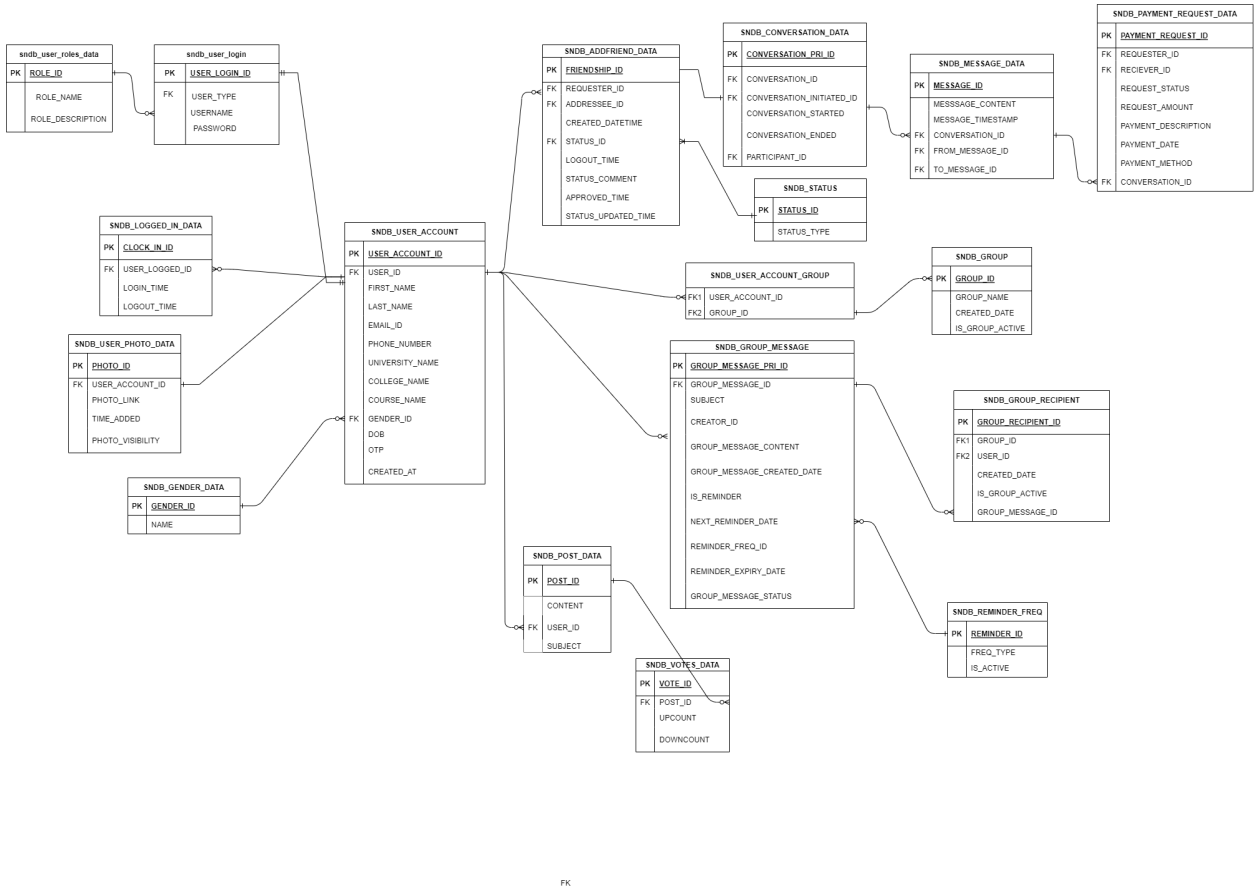
Each time a student log's in, students login history will be stored in the **SNDB_LOGGED_IN_DATA** table.
SNDB_GENDER_DATA table will contain the list of genders available in the database.
Students can upload their profile photo and the photo details and visibility details are maintained in the **SNDB_USER_PHOTO_DATA** 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.

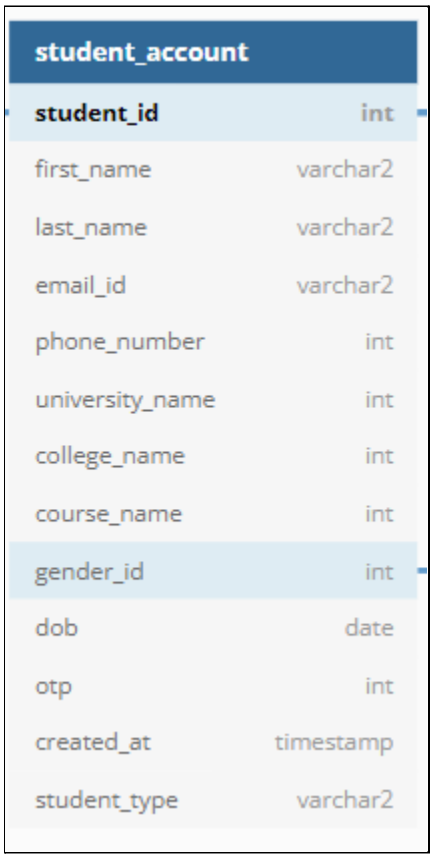


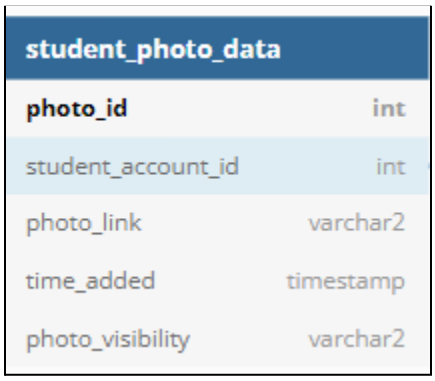
Relationships in ER Diagram- For reference

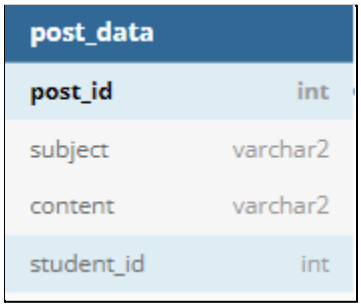
Analogies : **>** many-to-one; **<** one-to-many; **-** one-to-one

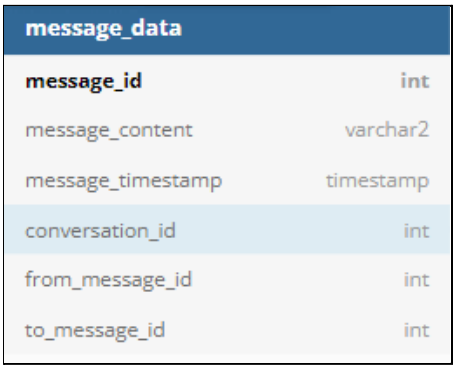
Ref:SNDB_USER_ROLES_DATA.role_name<SNDB_USER_LOGIN.user_type
Ref:SNDB_LOGGED_IN_DATA.USER_LOGGED_ID-SNDB_USER_ACCOUNT.user_id
Ref:SNDB_GENDER_DATA.gender_id < SNDB_USER_ACCOUNT.gender_id
Ref: SNDB_USER_ACCOUNT.user_id < SNDB_POST_DATA.user_id
Ref:SNDB_USER_PHOTO_DATA.user_account_id - SNDB_USER_ACCOUNT.user_id
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

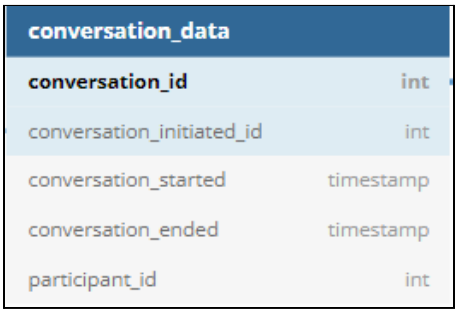
Ref: SNDB_MESSAGE_DATA.CONVERSATION_ID < SNDB_PAYMENT_REQUEST_DATA.CONVERSATION_ID
Ref:SNDB_USER_ACCOUNT.USER_ID < SNDB_USER_ACCOUNT_GROUP.USER_ACCOUNT_ID
Ref: SNDB_USER_ACCOUNT_GROUP.USER_ACCOUNT_ID < SNDB_GROUP.GROUP_ID
Ref:SNDB_USER_ACCOUNT.USER_ID < SNDB_GROUP_MESSAGE.CREATOR.ID
Ref:SNDB_GROUP_MESSAGE.GROUP_MESSAGE_ID < SNDB_GROUP_RECIPIENT.GROUP_MESSAGE_ID
Ref:SNDB_REMINDER_FREQ.REMINDER_ID < SNDB_GROUP_MESSAGE.REMINDER_FREQ_ID

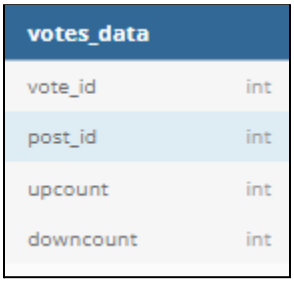


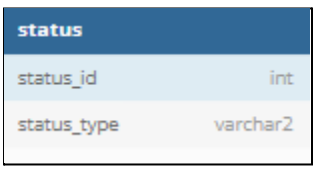



post_data
post_id int
subject varchar2
content varchar2
student_id int

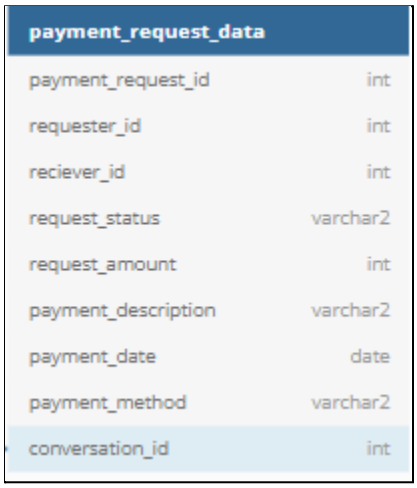

message_data
message_id int
message_content varchar2
message_timestamp timestamp
conversation_id int
from_message_id int
to_message_id int

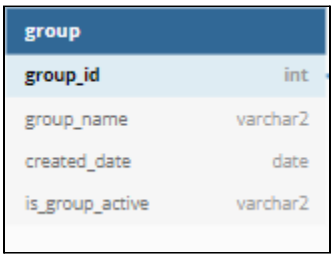

conversation_data
conversation_id int
conversation_initiated_id int
conversation_started timestamp
conversation_ended timestamp
participant_id int

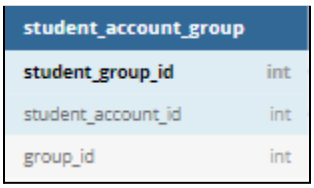

votes_data
vote_id int
post_id int
upcount int
downcount int

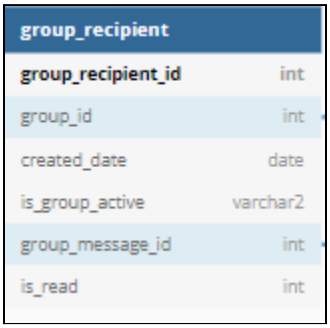

status
status_id int
status_type varchar2

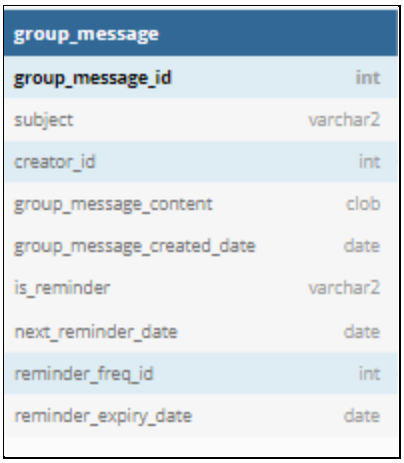

gender_data
gender_id int
name varchar2

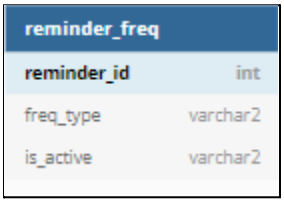

payment_request_data
payment_request_id int
requester_id int
reciever_id int
request_status varchar2
request_amount int
payment_description varchar2
payment_date date
payment_method varchar2
conversation_id int

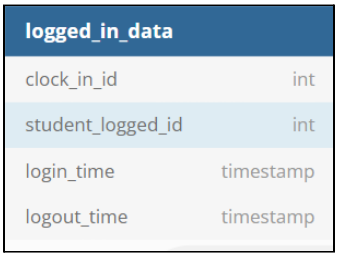

group
group_id int
group_name varchar2
created_date date
is_group_active varchar2

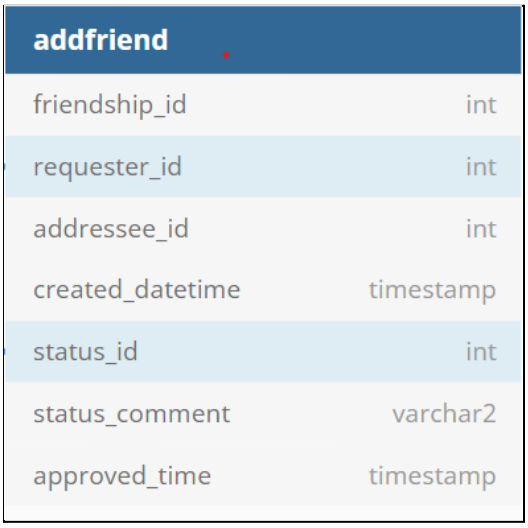

student_account_group
student_group_id int
student_account_id int
group_id int


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
next_reminder_date 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

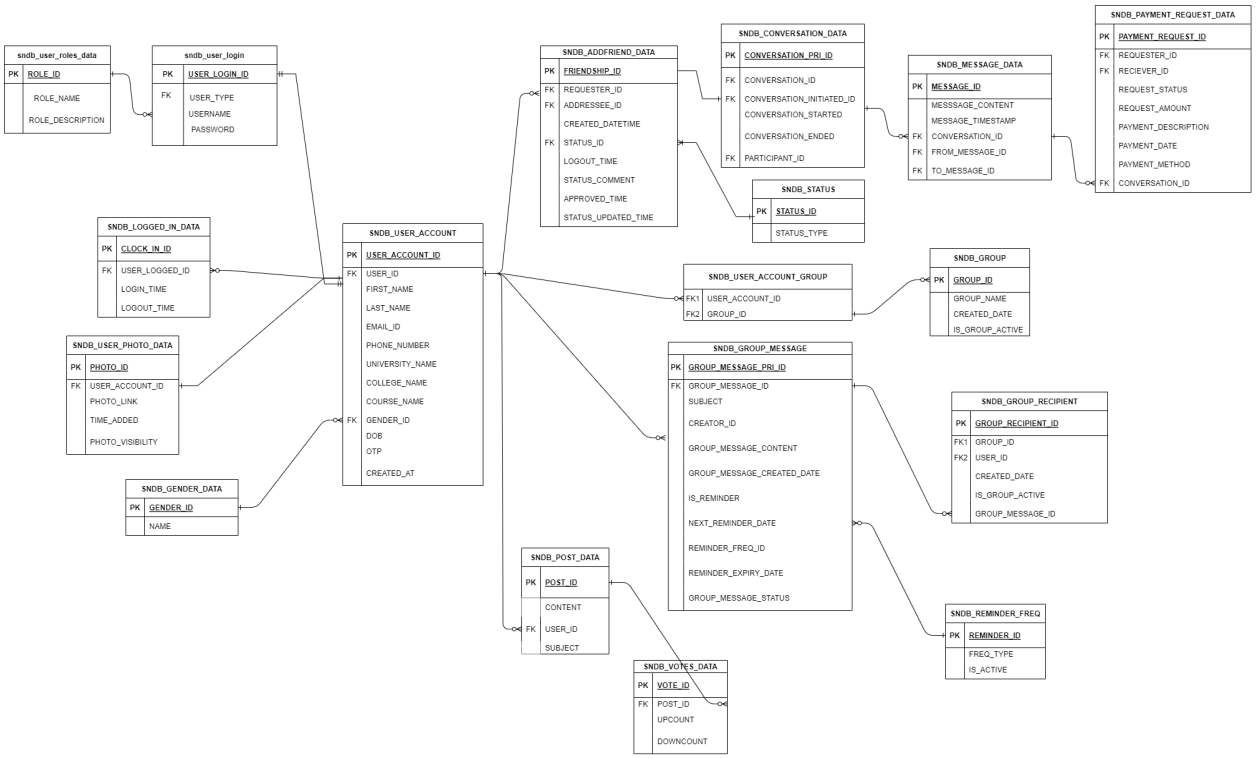
Sampada Dhole (NUID: **001526241**)
Meenakshi Vijay Kumbhar (NUID: **002922024**)
Vishnuprasad Thopucharla (NUID: **002926400**)
Nikhitha Dasi Srinivas (NUID: **002922544**)
Shravya Vura (NUID: **002927192**)

Business Rules:

Entity	Rules
User login	1) Admin or student can login using their username and password 2) User must have one role, either student or admin
User Account	1) User account should add their personal details like First Name, Last Name, Phone Number, Email Address 2) Student needs to be enrolled in a university and must provide university Name, College Name, Course Name 3) User can be uniquely identified by user_id 4) All the students in the user_account database are considered as verified students.
Post Data	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 friends 3) Students can dislike or like the posts
Add Friend	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	1) Student can see the upcount and downcount values of the post 2) Students can vote posts of other students even if they are not friends
Message Data	1) One on one conversation data will be stored in this.

	2) Message data will be identified by the primary key message_id
Payment Request Data	1) If the payment request is accepted, bill details are displayed.
Group message data	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.
Group Recipient Data	1) Group will inactivate when the is_group_active flag is set to false. 2) Group creator's data will be stored in this.
Reminder Frequency Data	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.
Photo data	1) Student can view/update their own profile photo.

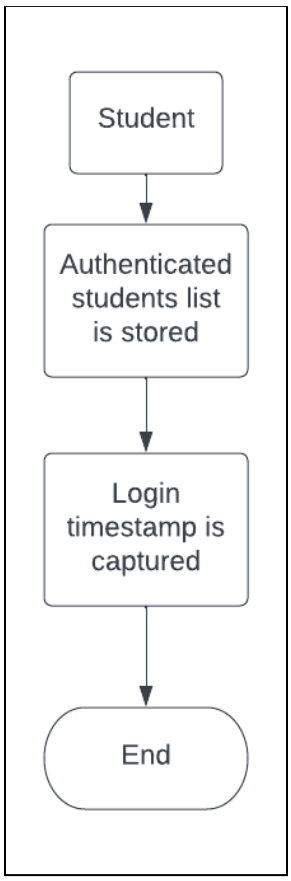
Revised ER diagram:



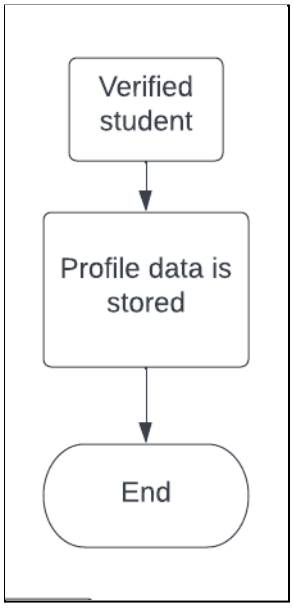
FK

DFD Diagram:

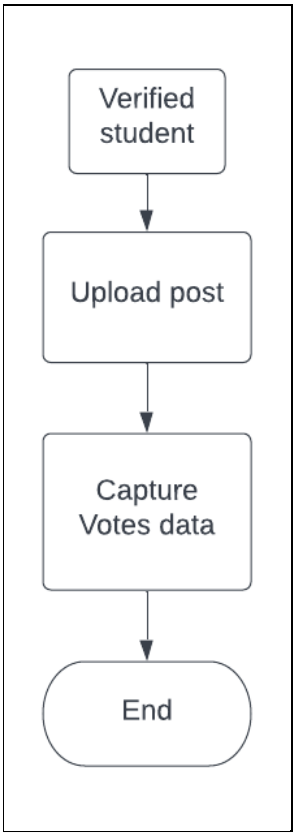
1. User Account Login Flow:



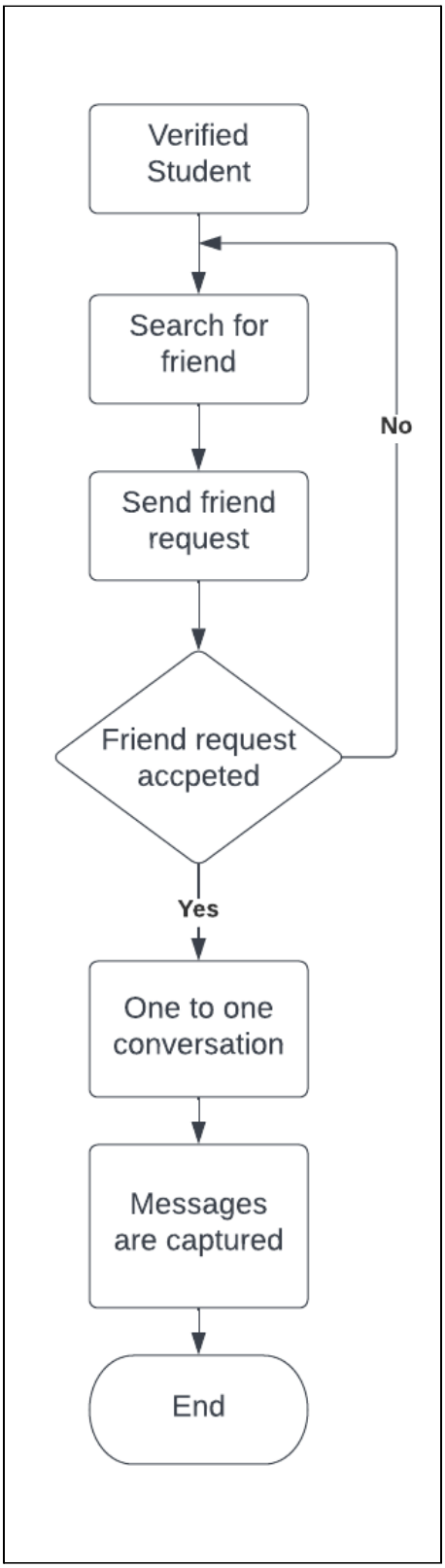
2. Profile Picture Flow



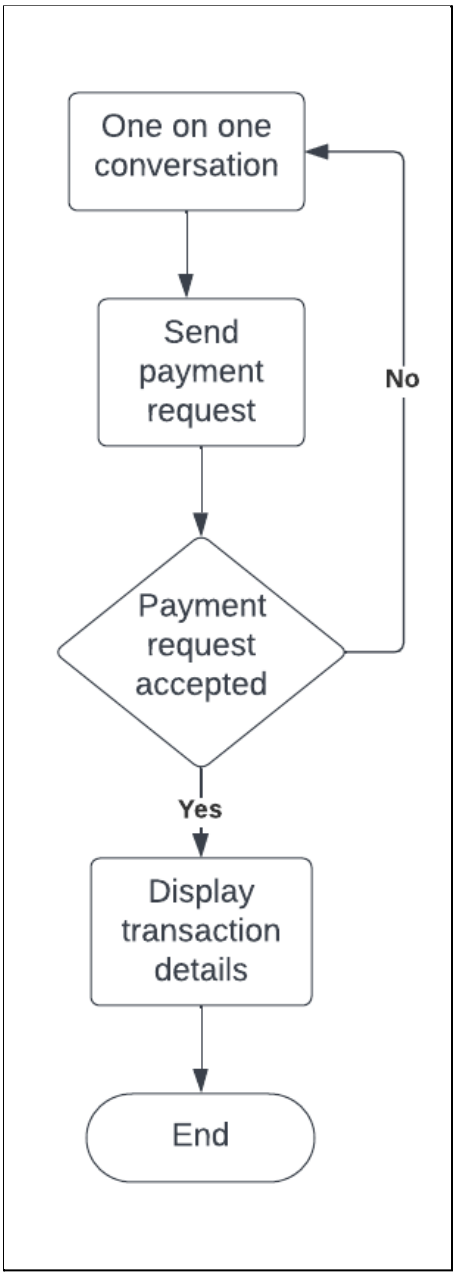
3. Post and Votes Flow



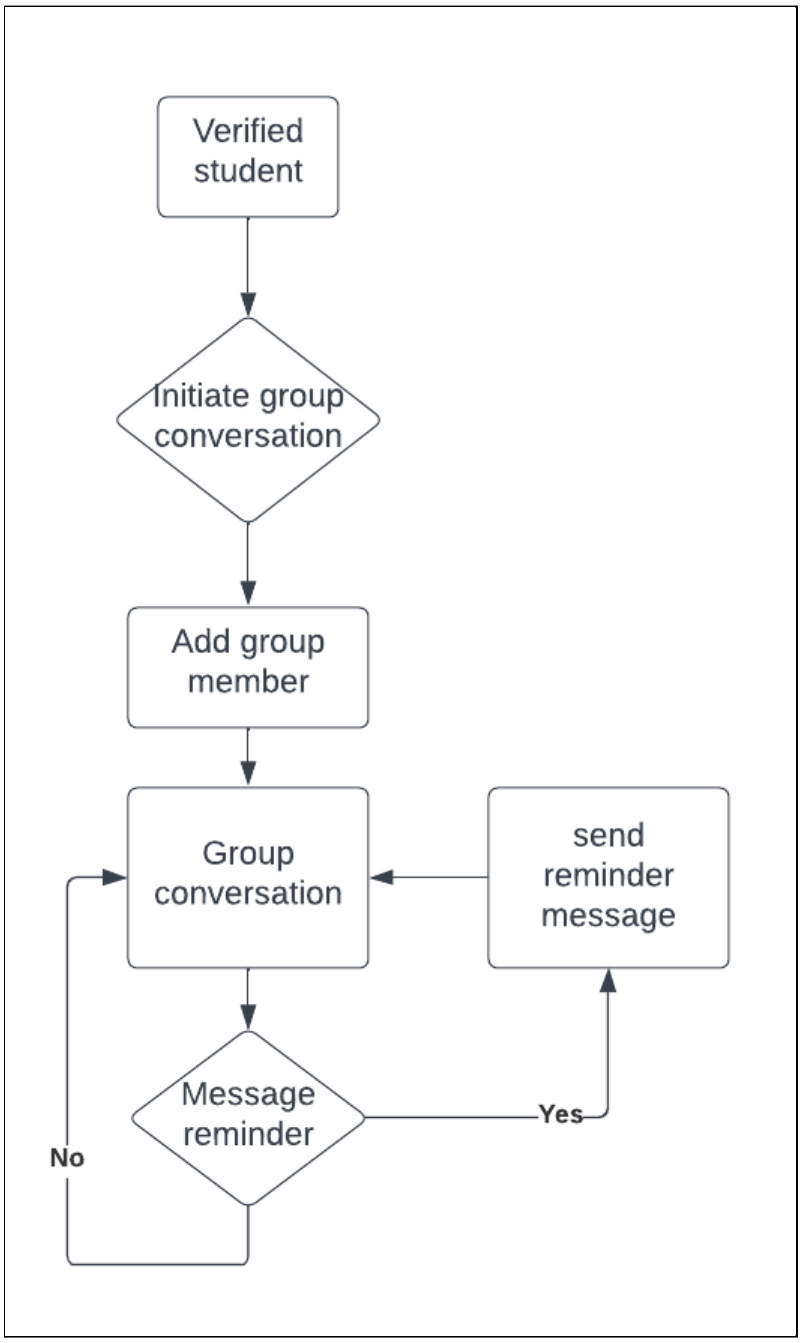
4. Friend Request and One on One Conversation Flow:



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;
/

DECLARE
    p_user_login NUMBER;
    user_login_exsists 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_exsists;
    END IF;
EXCEPTION
    WHEN user_login_exsists 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_exsists 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_exsists ;
   END IF;
EXCEPTION
   WHEN gender_data_table_exsists 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_exsists 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_exsists ;
   END IF;
EXCEPTION
   WHEN user_photo_data_table_exsists 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;
/

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;
```

```
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;
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,
     Is_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_exsists;
    END IF;
EXCEPTION
    WHEN reminder_freq_exsists 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_exsists 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,
```

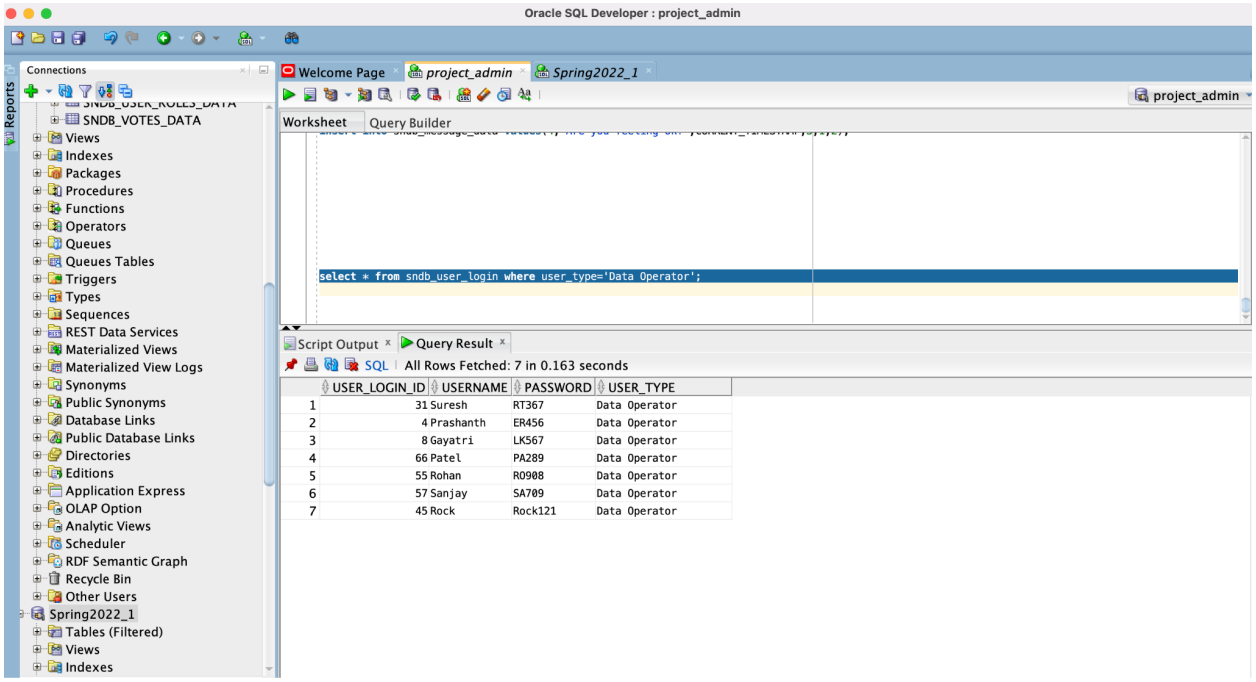
```
      group_message_created_date date NOT NULL,
      Is_reminder varchar2(10) NOT NULL,
      Next_reminder_date date NOT NULL,
      reminder_freq_id number NOT NULL,
      reminder_expiry_date date NOT NULL,
      group_message_status varchar2(10),
CONSTRAINT group_message_id_pk PRIMARY KEY (group_message_PRI_id ),
  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:

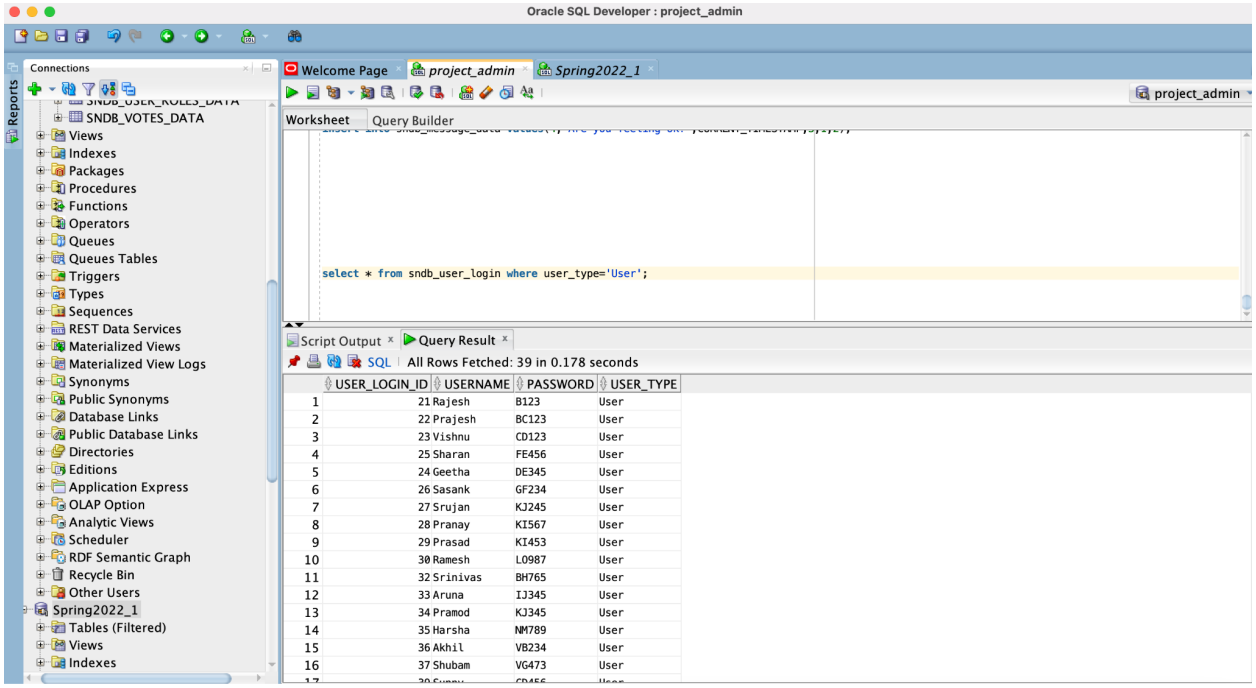
Scenario I:

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

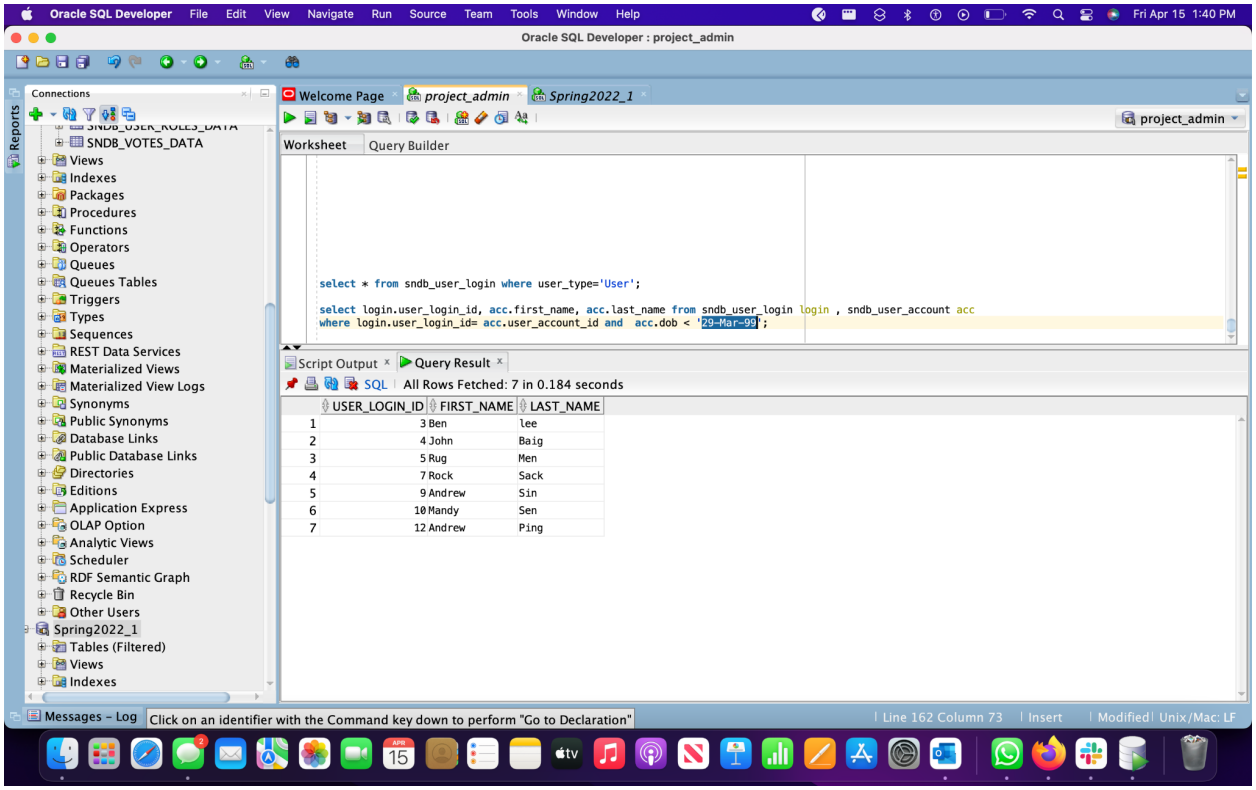
Script: select * from sndb_user_login where user_type='Data Operator';



Here, we are checking the user role

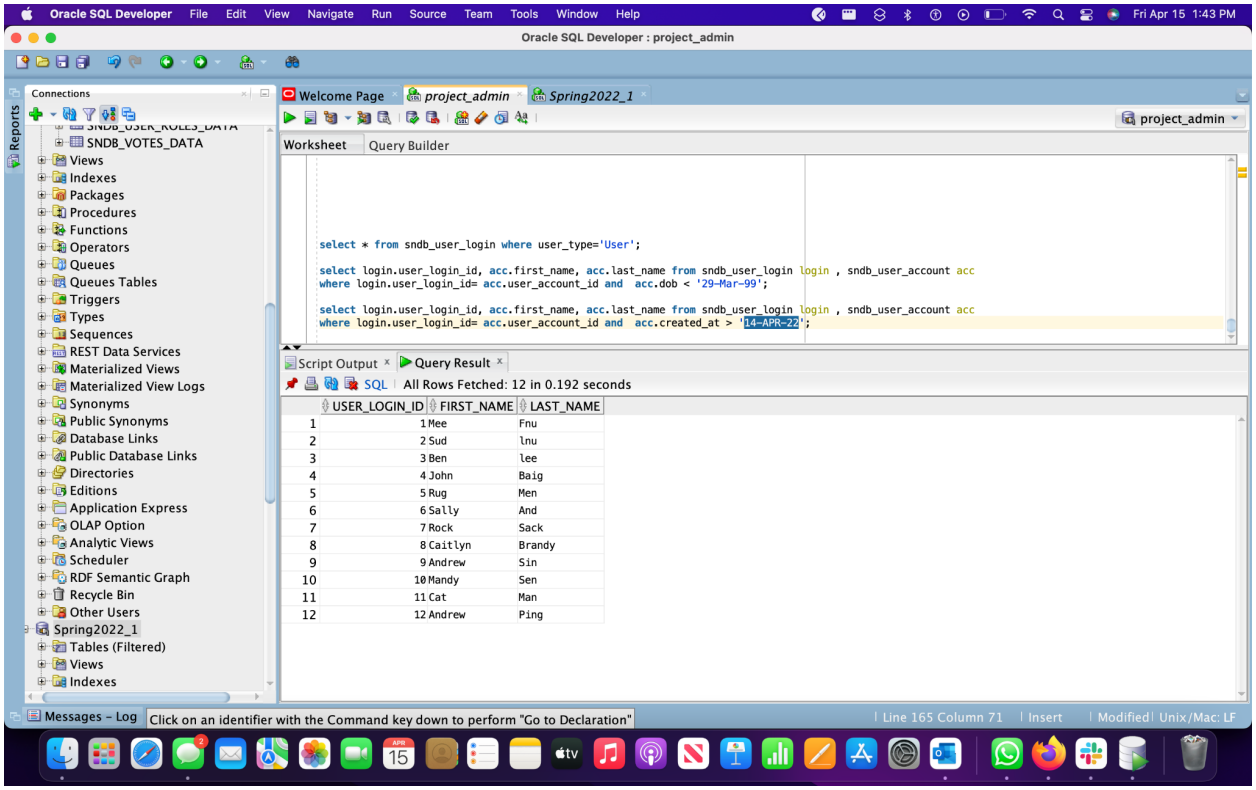


To get user_id, first_name and last_name where dob of an user is less than 29-Mar-99
Script: select login.user_login_id, acc.first_name, acc.last_name from sndb_user_login login , sndb_user_account acc
where login.user_login_id= acc.user_account_id and acc.dob < '29-Mar-99';



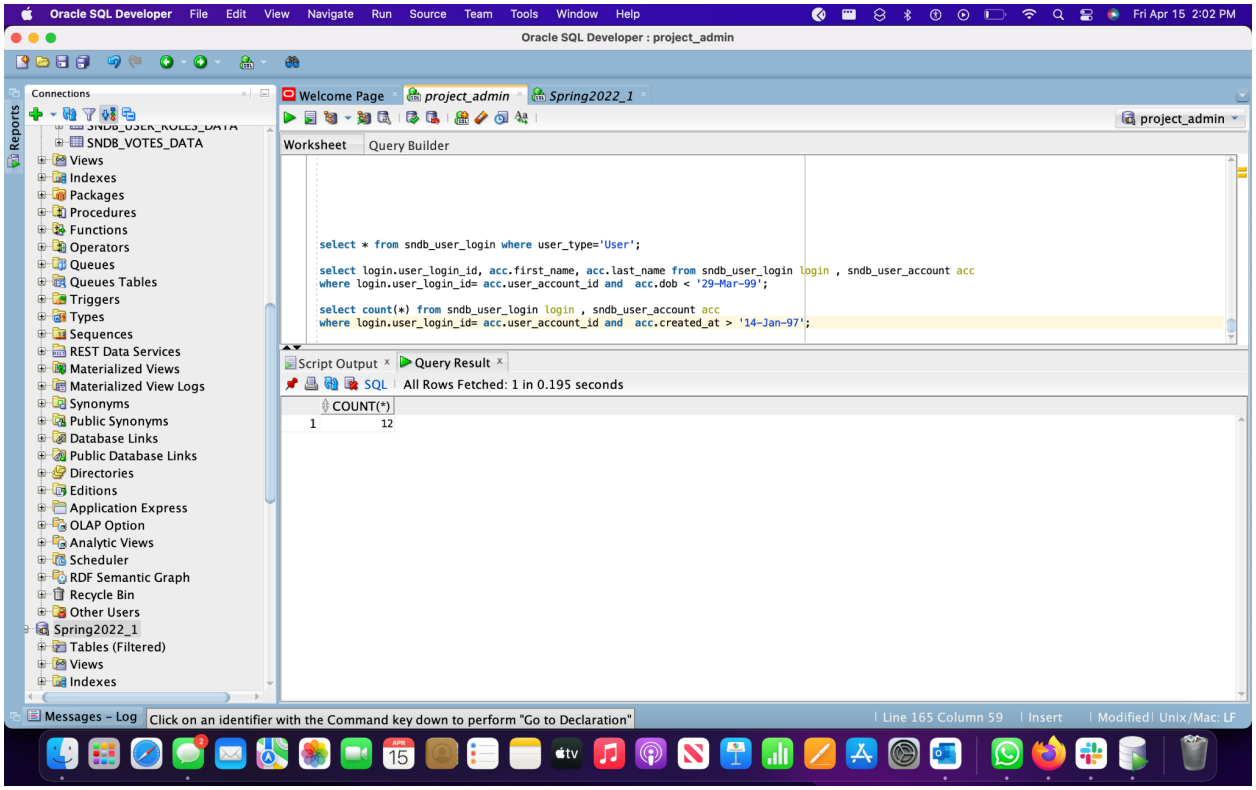
The users who joined the application in April

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';



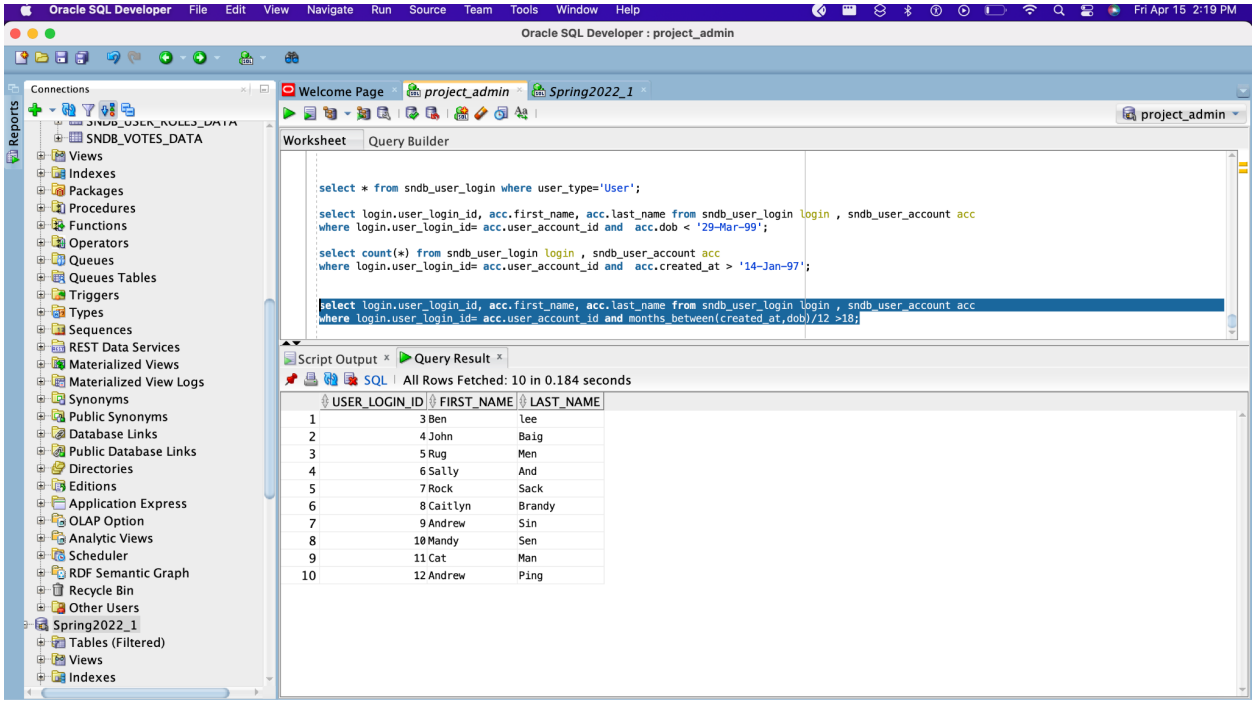
Count of users who joined the application after Jan-97

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';



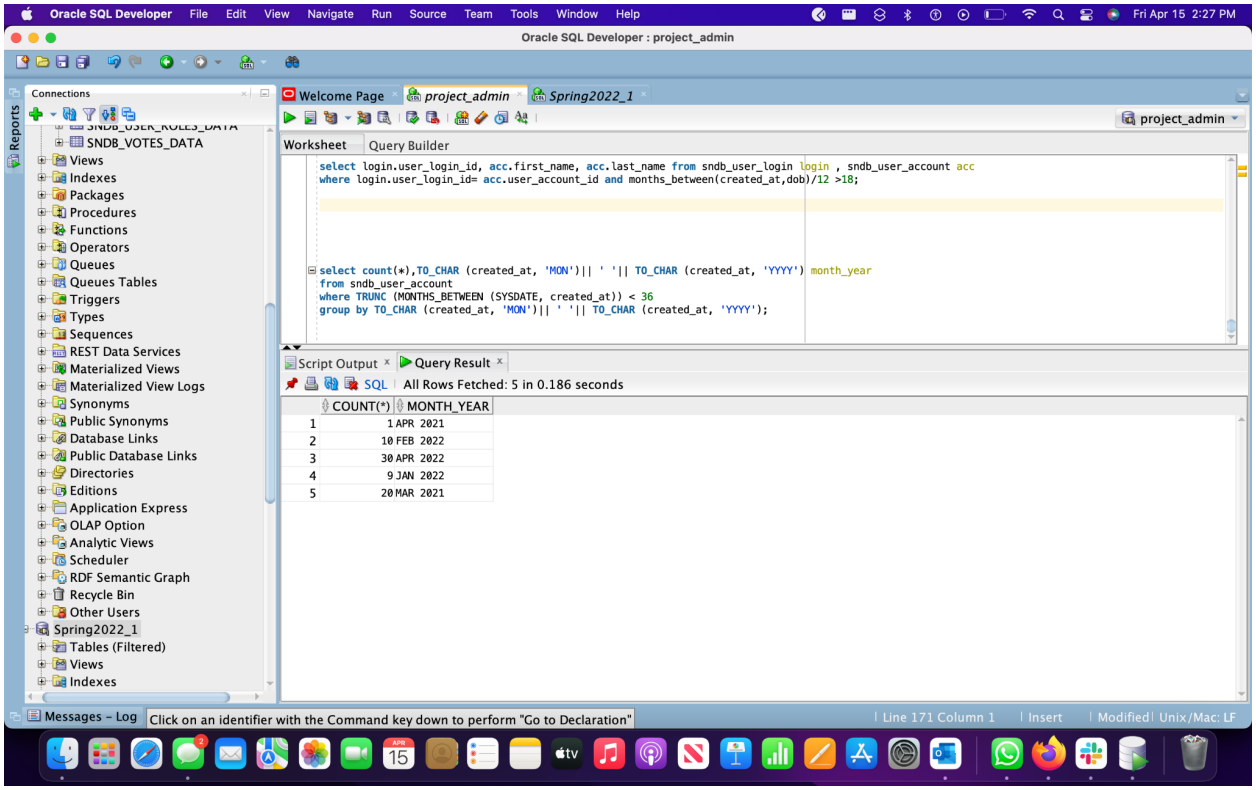
Users who are elder than 18 years

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;



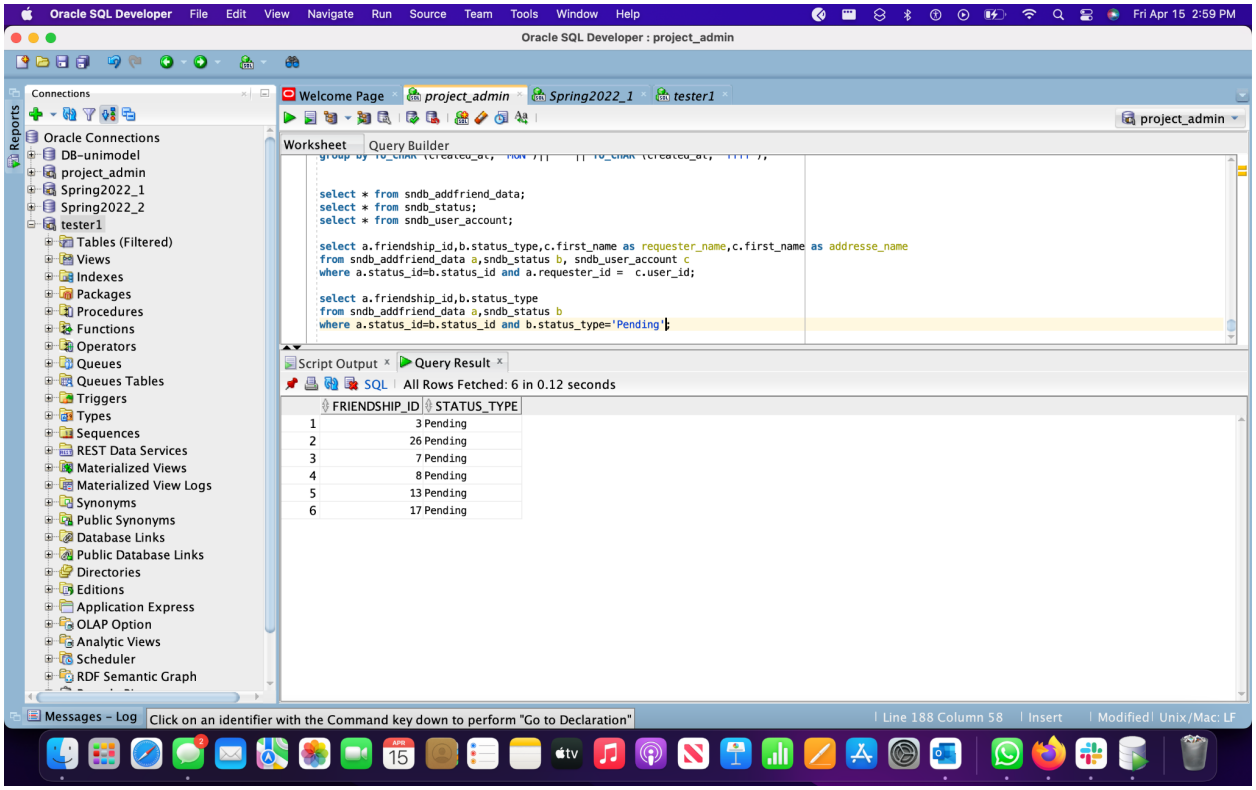
To get the count of users who joined for the last three years month wise

```
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');
```



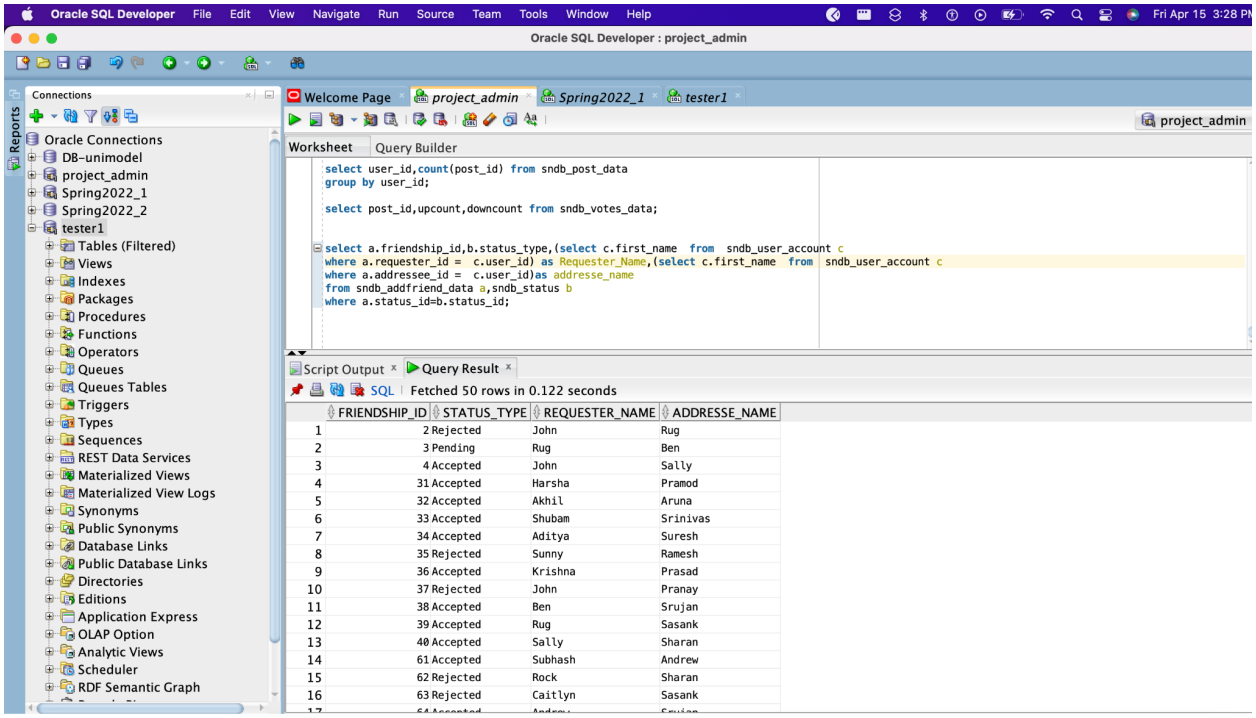
To get the records of users whose friendship request is pending

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';



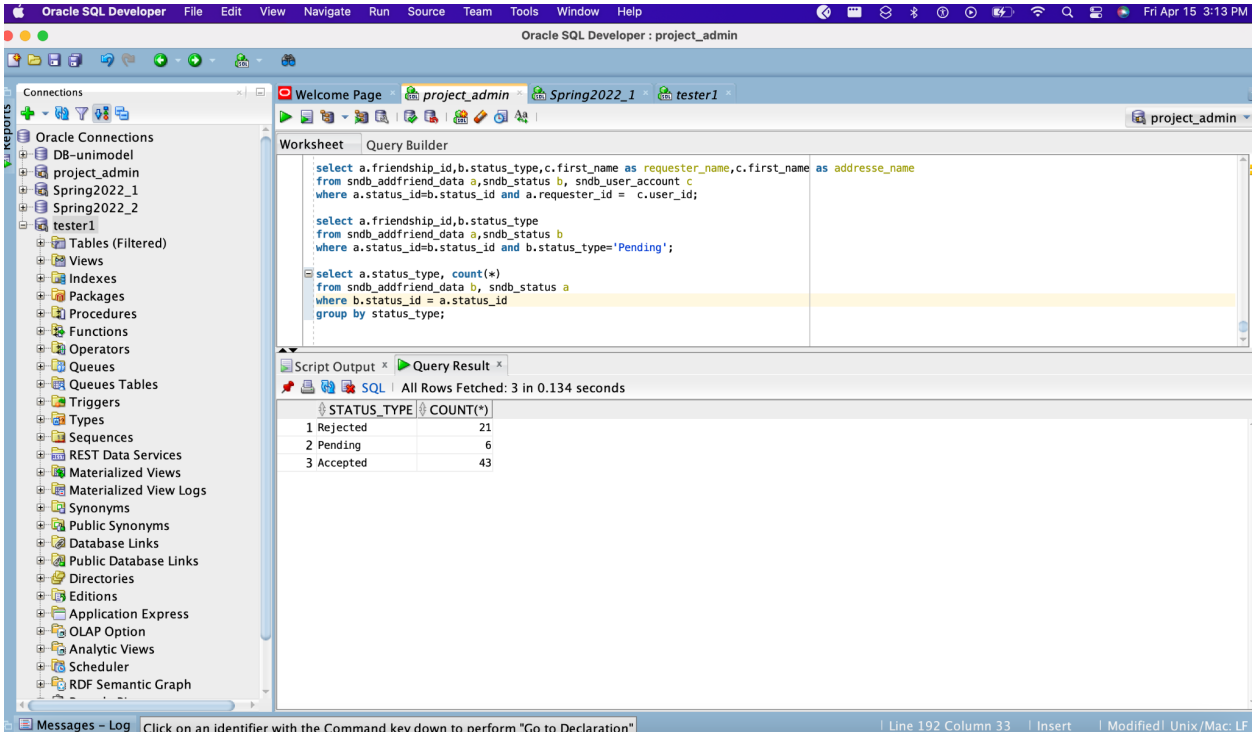
To get the requester name, addressee name and request status

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 addresse_name
from sndb_addfriend_data a,sndb_status b
where a.status_id=b.status_id;



Count of pending, accepted and rejected requests

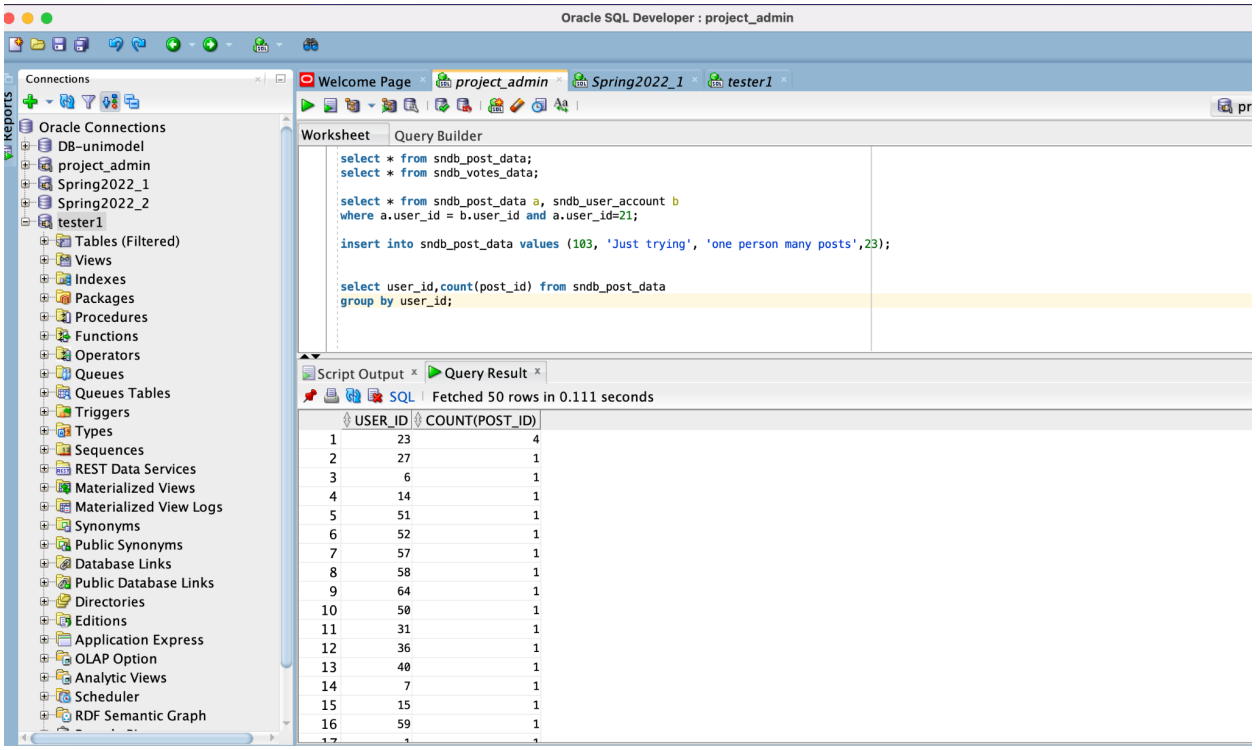
Script: select a.status_type, count(*)
from sndb_addfriend_data b, sndb_status a
where b.status_id = a.status_id
group by status_type;



Scenario 2:

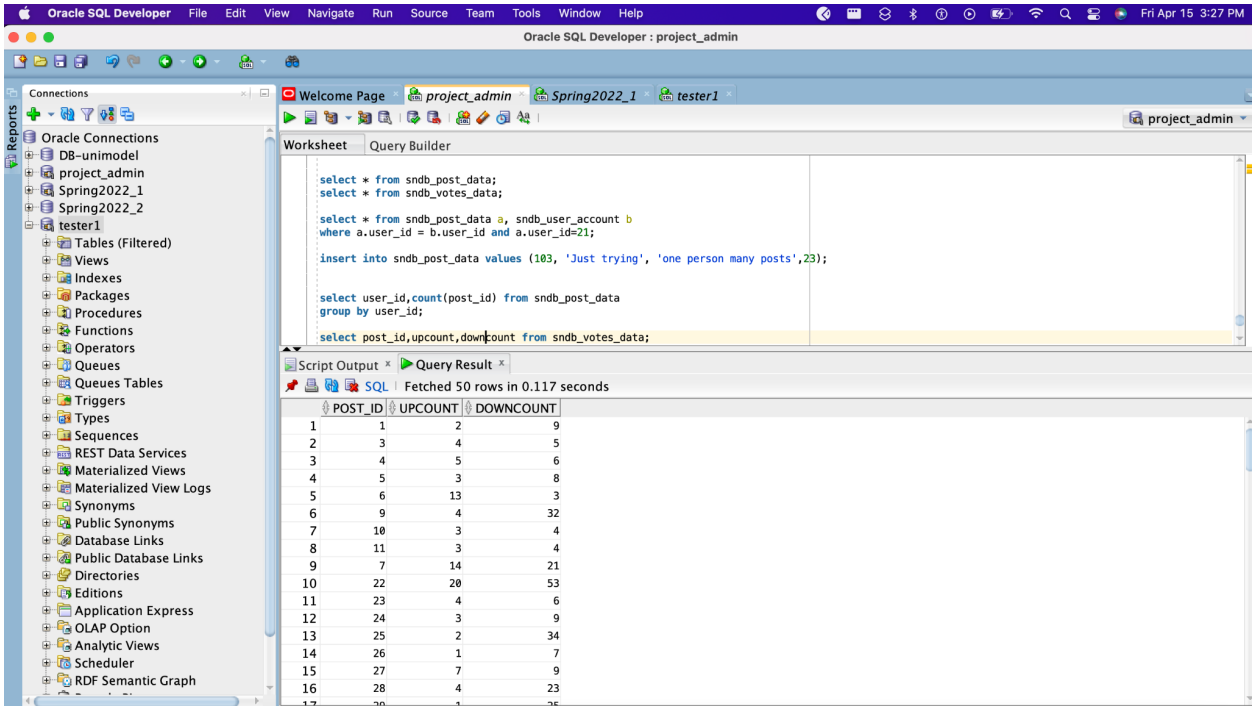
Number of posts by each user

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



Likes and dislikes on each post

Script: select post_id,upcount,downcount from sndb_votes_data;

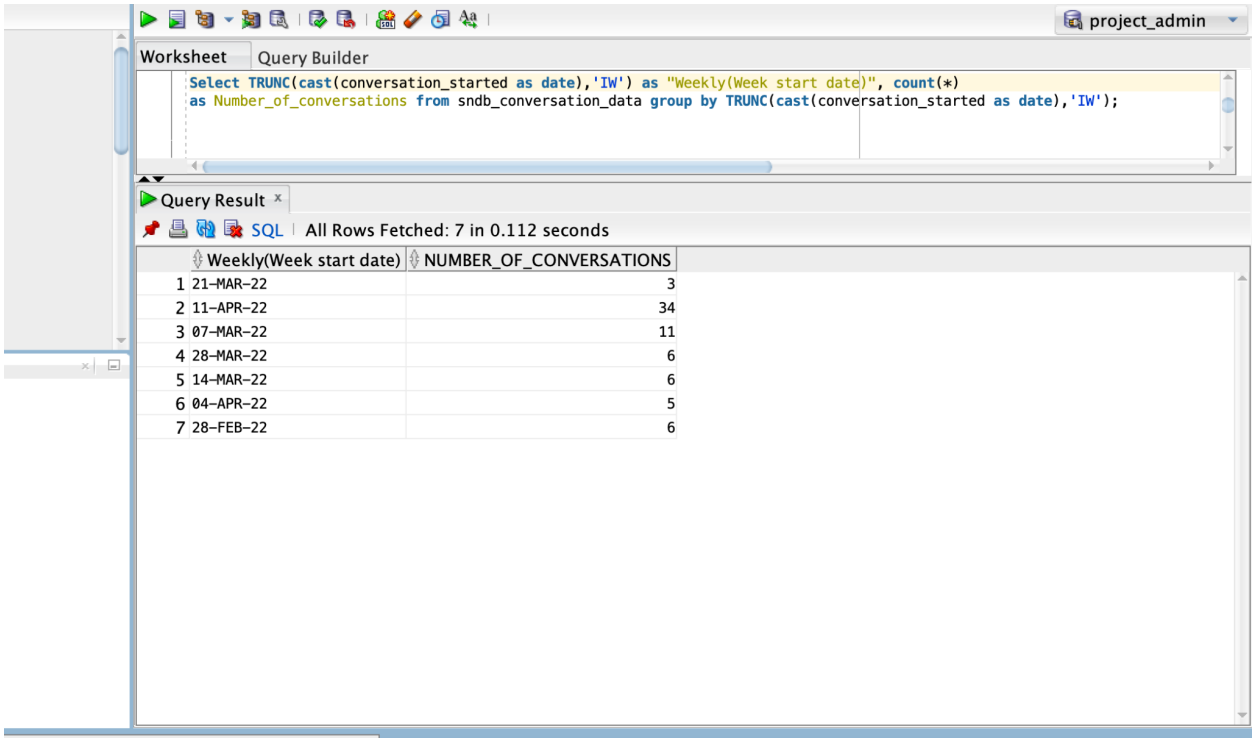


SCENARIO 3:

Number of conversations started in each week

Script:

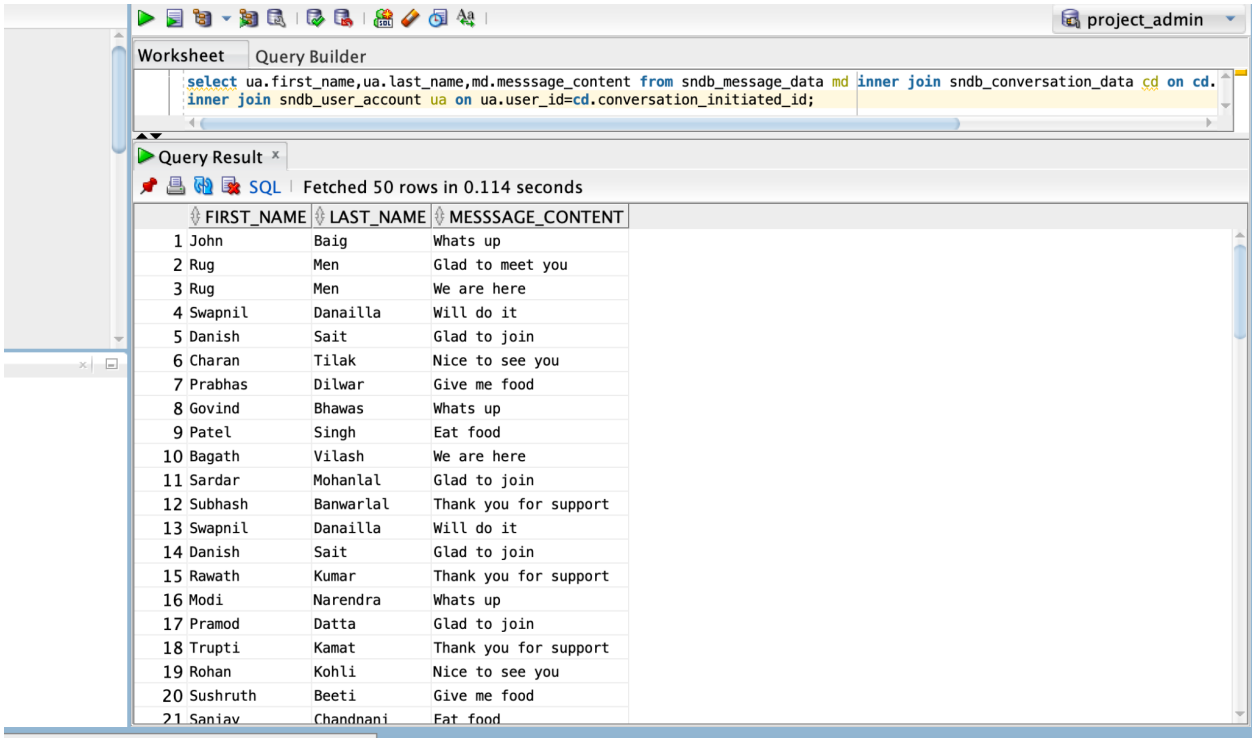
Select TRUNC(cast(conversation_started as date),'IW') as "Weekly(Week start date)", count(*) as Number_of_conversations from sndb_conversation_data group by TRUNC(cast(conversation_started as date),'IW');



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;
```

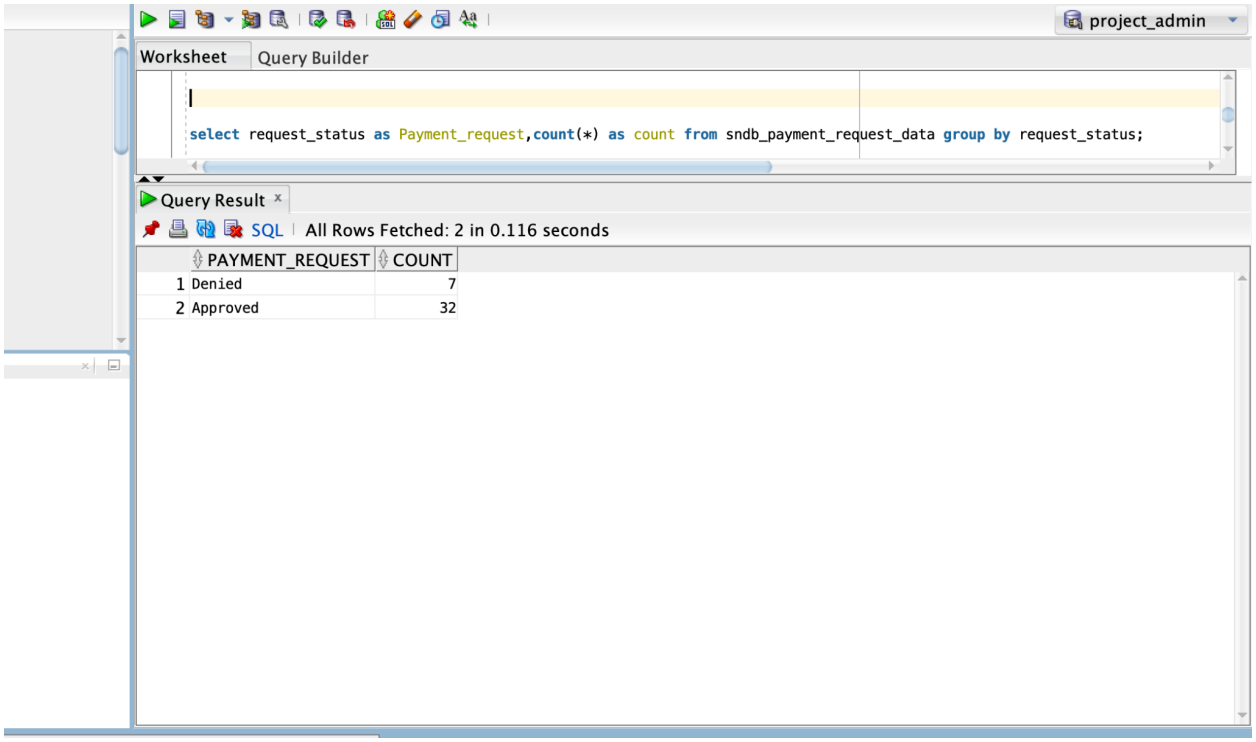


	FIRST_NAME	LAST_NAME	MESSSAGE_CONTENT
1	John	Baig	Whats up
2	Rug	Men	Glad to meet you
3	Rug	Men	We are here
4	Swapnil	Danailla	Will do it
5	Danish	Sait	Glad to join
6	Charan	Tilak	Nice to see you
7	Prabhas	Dilwar	Give me food
8	Govind	Bhawas	Whats up
9	Patel	Singh	Eat food
10	Bagath	Vilash	We are here
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	Sanjay	Chandnani	Eat food

Number of Payment Requests Approved/Denied

Script:

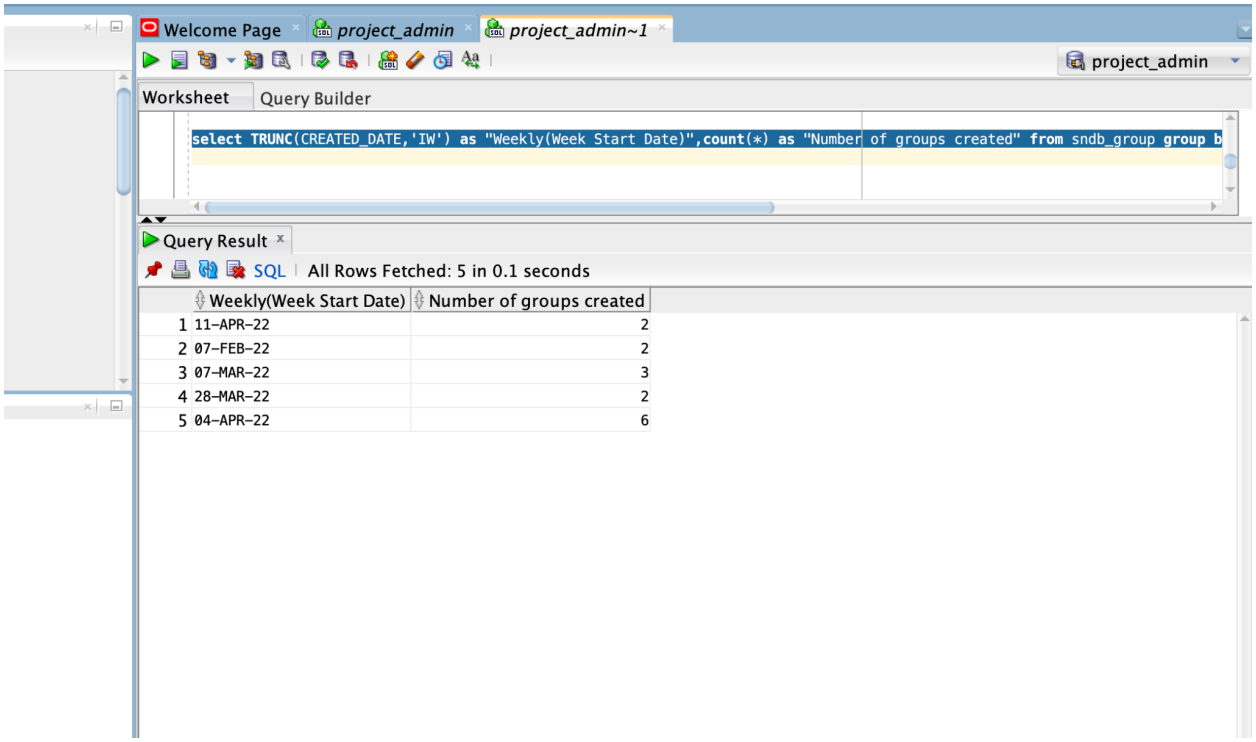
```
select request_status as Payment_request,count(*) as count from sndb_payment_request_data
group by request_status;
```



SCENARIO 4

Number of groups created in the week

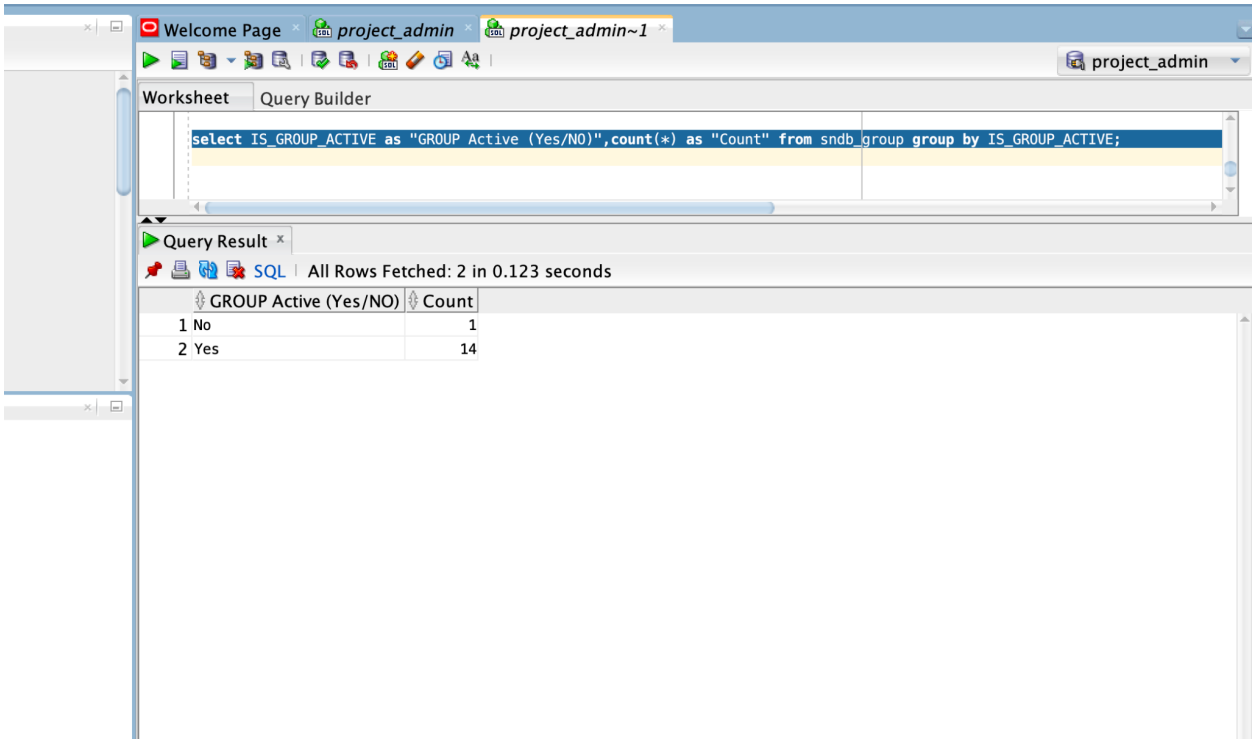
Script:
select TRUNC(CREATED_DATE,'IW') as "Weekly(Week Start Date)",count(*) as "Number of groups created" from sndb_group group by TRUNC(CREATED_DATE,'IW');



Number of active and inactive groups

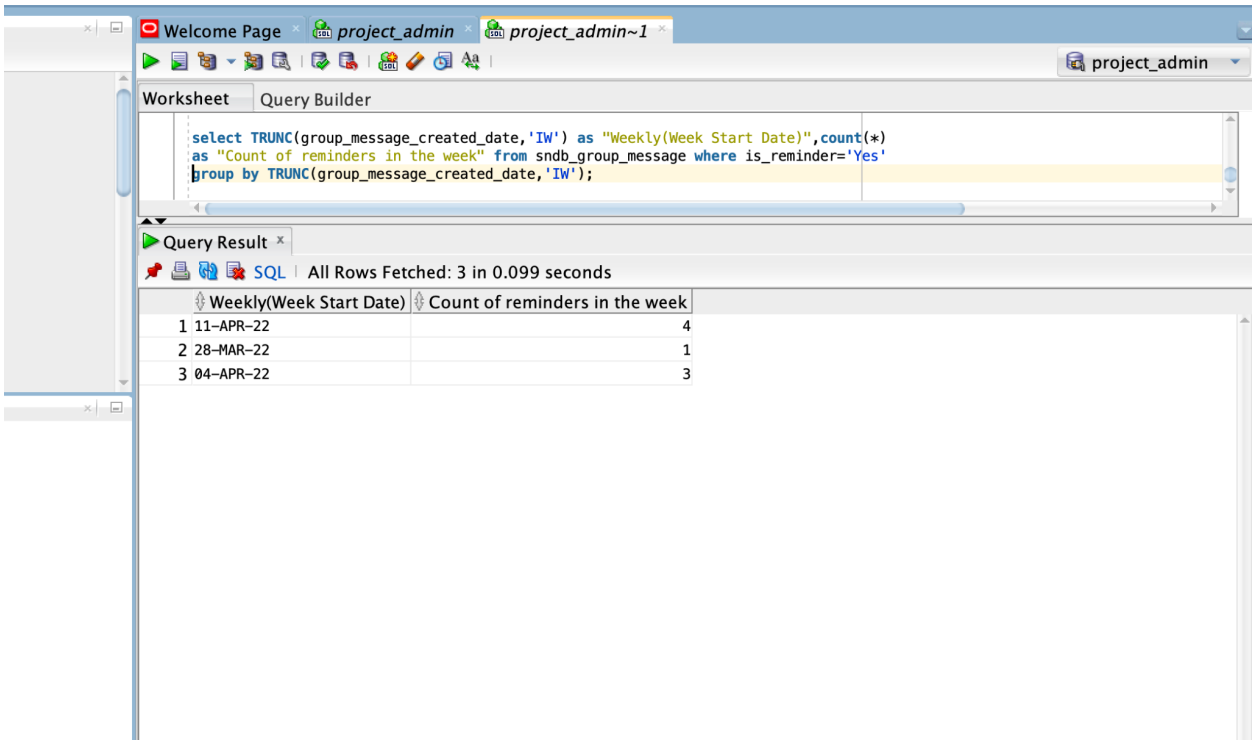
Script:

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



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 is_reminder='Yes' group by TRUNC(group_message_created_date,'IW');



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');
```

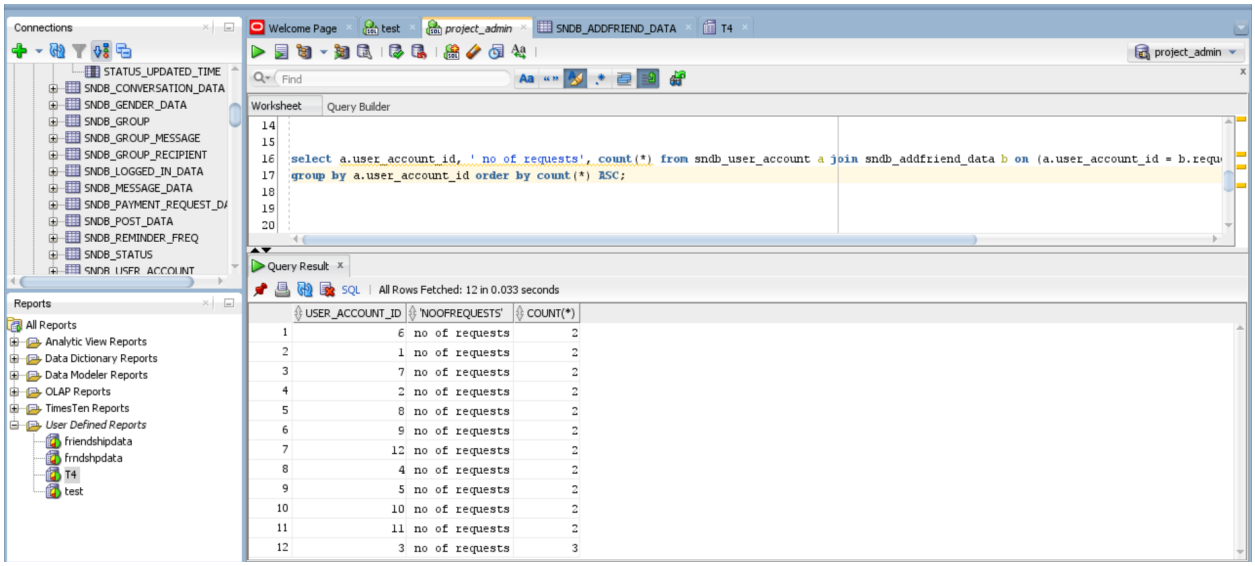
```
insert into sndb_user_photo_data values(3,5,'http://tinyurl/you',TO_TIMESTAMP('06-MAR-22
05.09.59', 'DD-MON-YY HH24:mi:ss'),'No');

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');
```

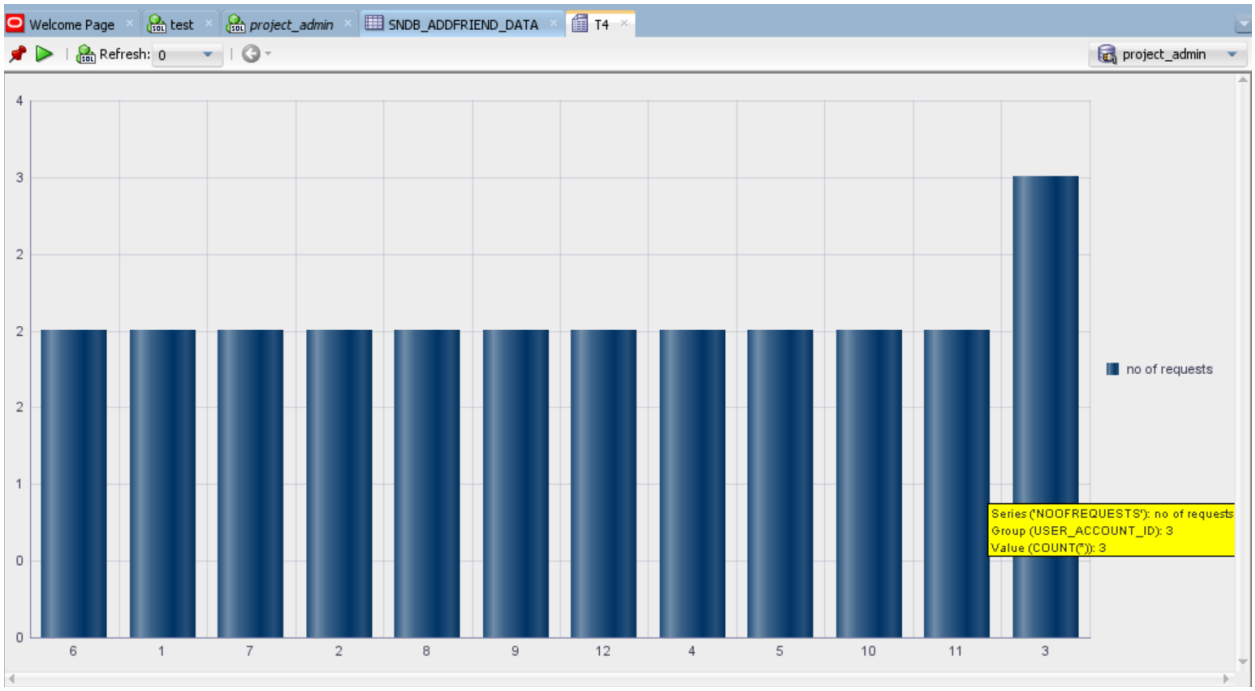
PROJECT 5 (Team 5)

REPORTS:

1. NO OF FRIEND REQUESTS:

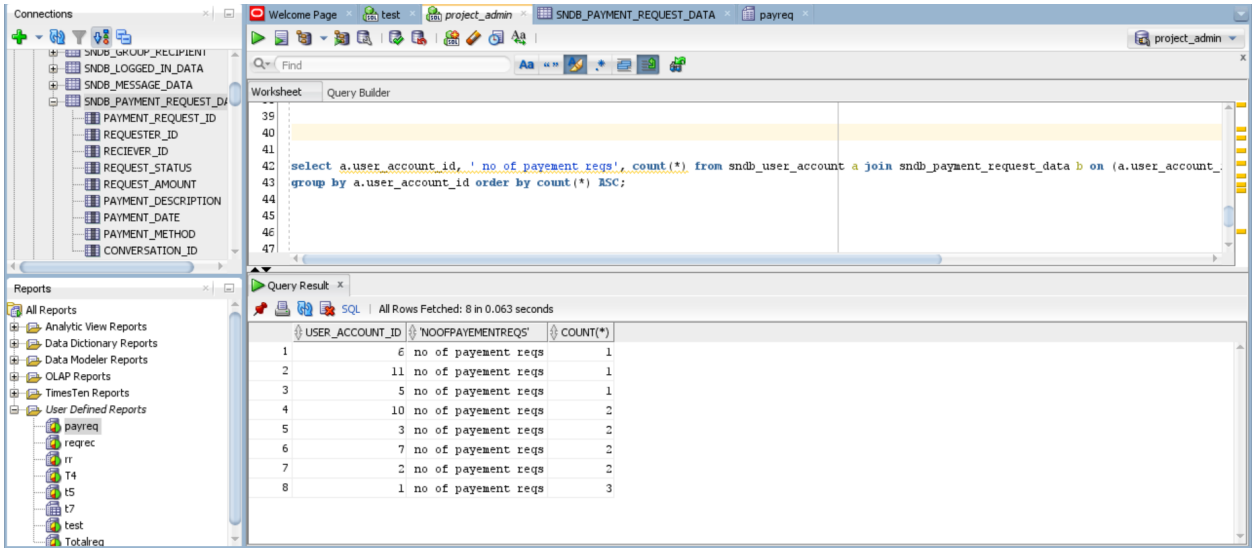
```
select a.user_account_id, ' no of requests', count(*) from sndb_user_account a join
sndb_addfriend_data b on (a.user_account_id = b.requester_id or a.user_account_id =
b.addressee_id)
group by a.user_account_id order by count(*) ASC;
```

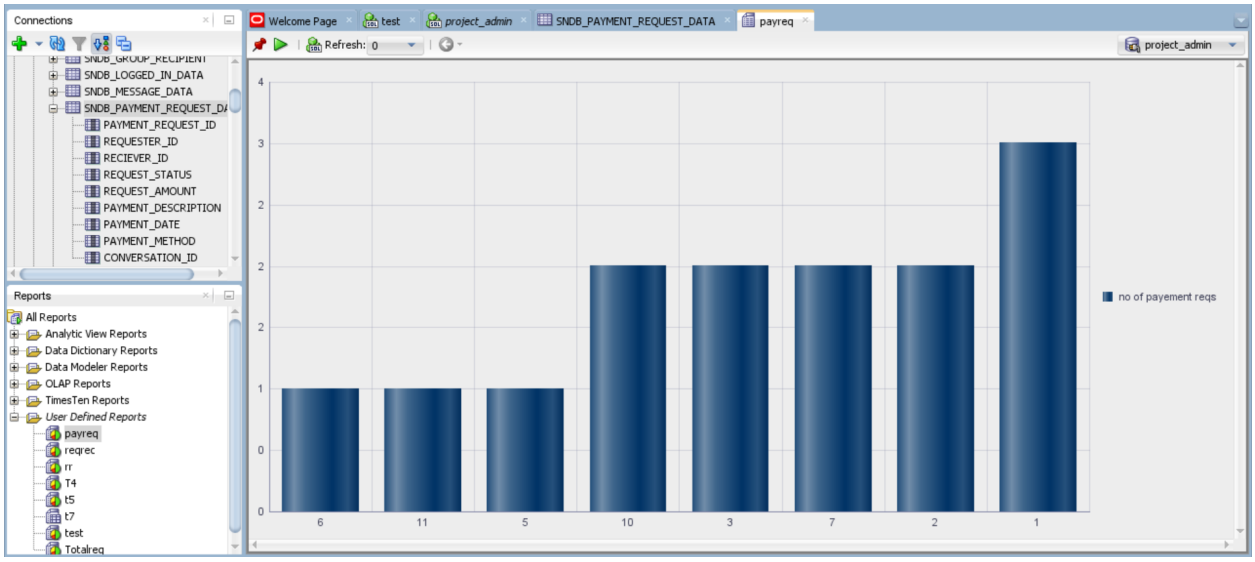




2. NO OF PAYMENT REQUESTS:

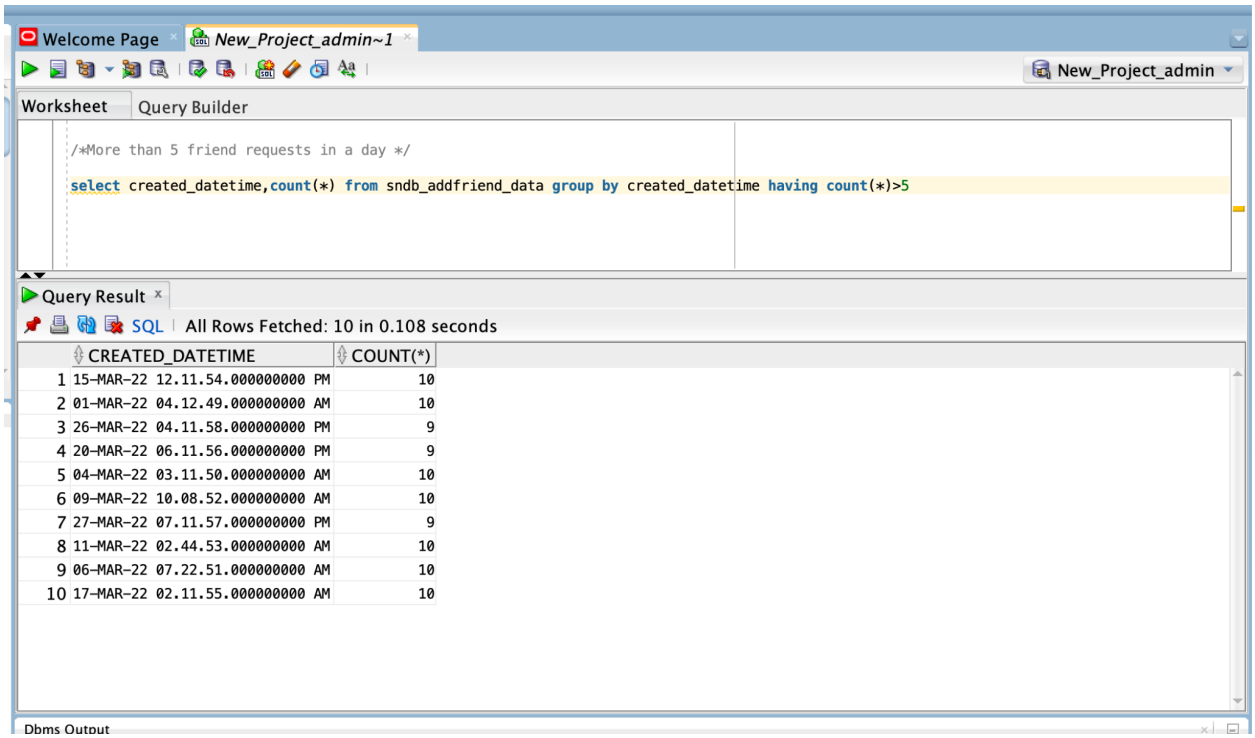
select a.user_account_id, ' no of payement reqs', count(*) from sndb_user_account a join sndb_payment_request_data b on (a.user_account_id = b.requester_id or a.user_account_id = b.reciever_id)
group by a.user_account_id order by count(*) ASC;


Connections
SNDB_LOGGED_IN_DATA
SNDB_MESSAGE_DATA
SNDB_PAYMENT_REQUEST_DA
PAYMENT_REQUEST_ID
REQUESTER_ID
RECIEVER_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
Welcome Page
test
project_admin
SNDB_PAYMENT_REQUEST_DATA
payreq
project_admin
Find
Worksheet
Query Builder
39
40
41
42 select a.user_account_id, ' no of payement reqs', count(*) from sndb_user_account a join sndb_payment_request_data b on (a.user_account_
43 group by a.user_account_id order by count(*) ASC;
44
45
46
47
Query Result
SQL
All Rows Fetched: 8 in 0.063 seconds
USER_ACCOUNT_ID
'NOOFPAYEMENTREQS'
COUNT(*)
1 6 no of payement reqs 1
2 11 no of payement reqs 1
3 5 no of payement reqs 1
4 10 no of payement reqs 2
5 3 no of payement reqs 2
6 7 no of payement reqs 2
7 2 no of payement reqs 2
8 1 no of payement reqs 3


Connections
SNDB_LOGGED_IN_DATA
SNDB_MESSAGE_DATA
SNDB_PAYMENT_REQUEST_DA
PAYMENT_REQUEST_ID
REQUESTER_ID
RECIEVER_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
Welcome Page
test
project_admin
SNDB_PAYMENT_REQUEST_DATA
payreq
Refresh:
0
project_admin
4
3
2
2
1
0
6
11
5
10
3
7
2
1
no of payement reqs

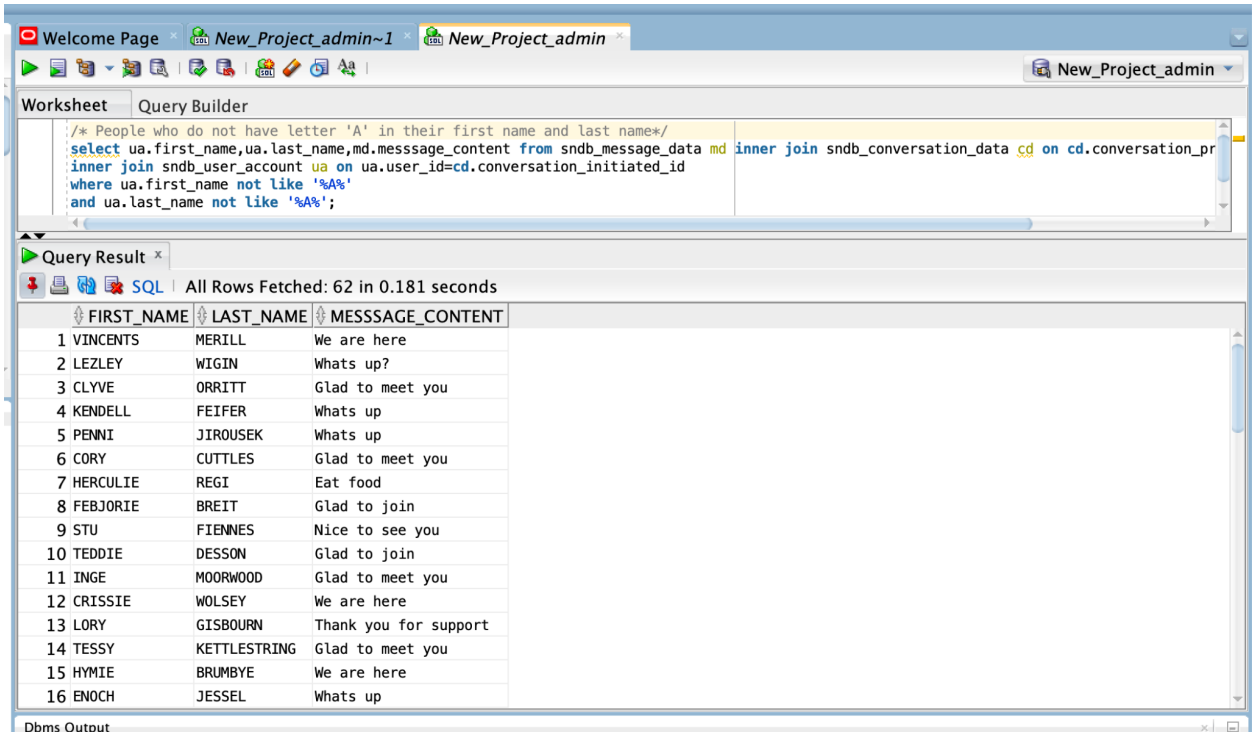
3. More than 5 friend requests in a day:

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



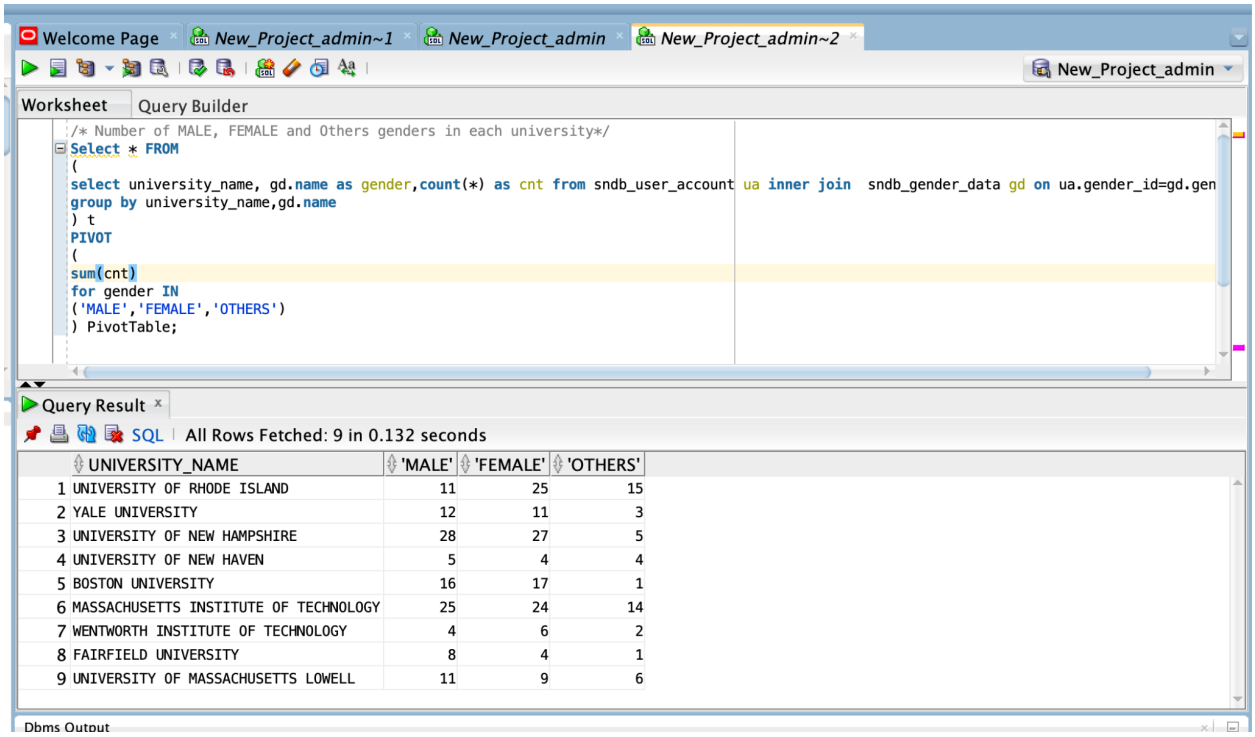
4. People who do not have letter 'A' in their first name and last name

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
where ua.first_name not like '%A%'
and ua.last_name not like '%A%';



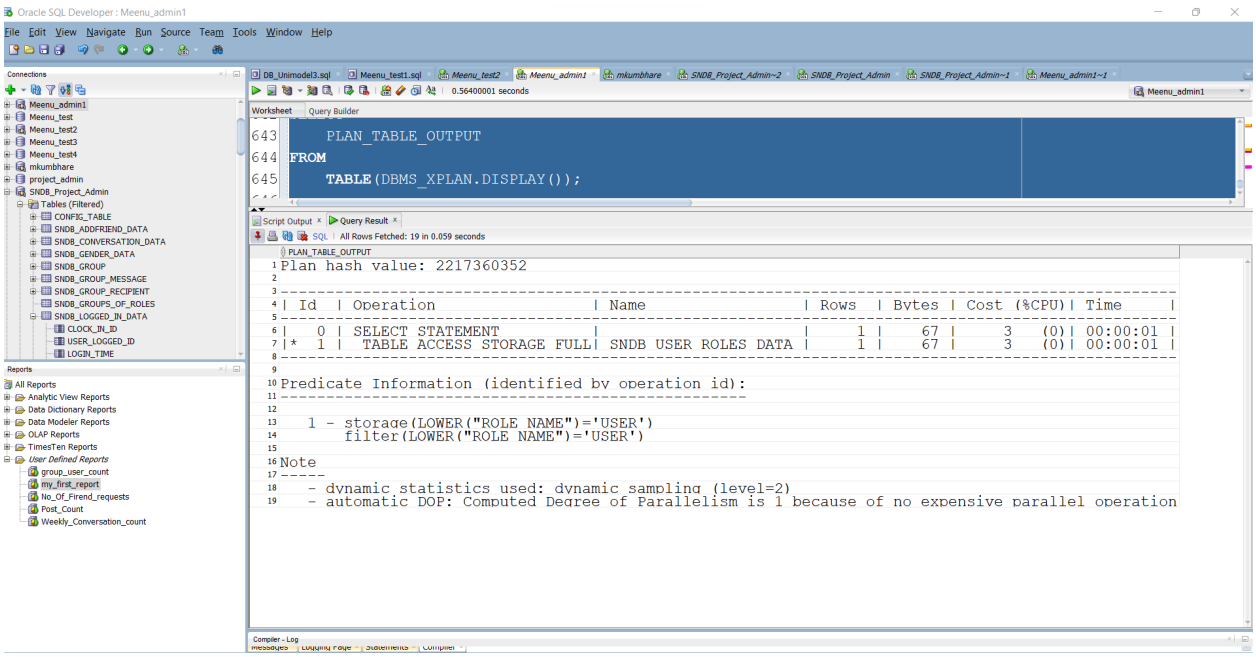
5. Number of MALE, FEMALE and Others genders in each university

```
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;
```


Welcome Page
New_Project_admin~1
New_Project_admin
New_Project_admin~2
New_Project_admin
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
SQL
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

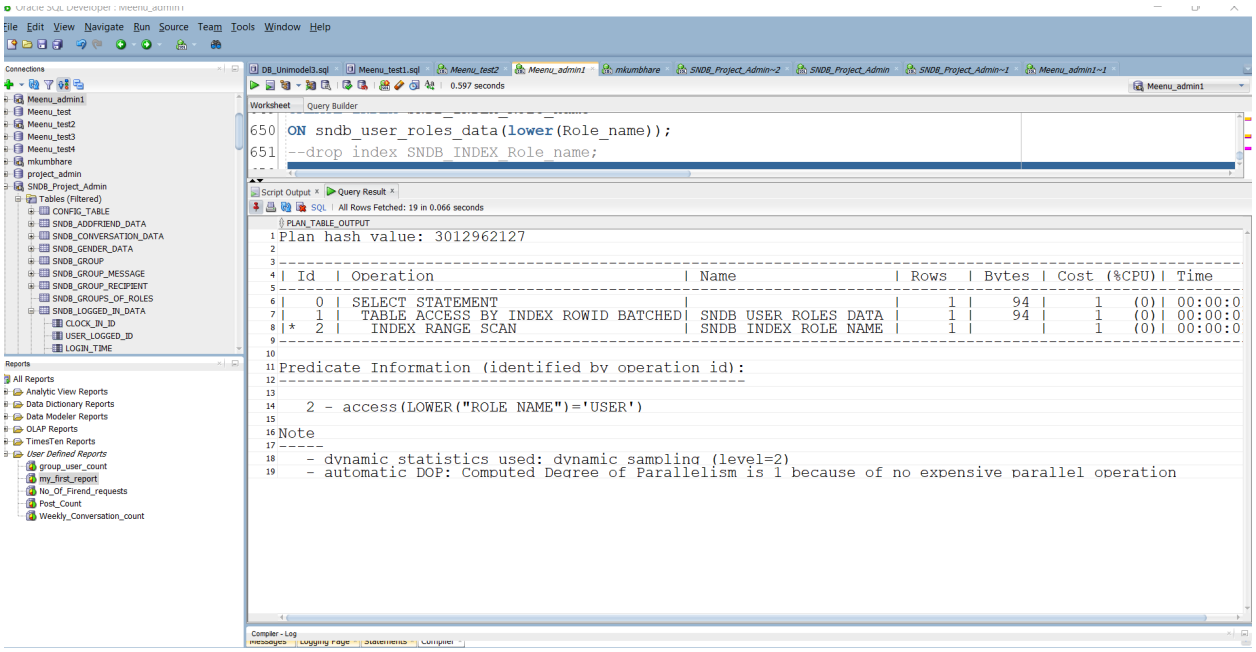
--WIthout INDEXES

```
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
Tables (Filtered)
CONFIG_TABLE
SNDB_ADDFRIEND_DATA
SNDB_CONVERSATION_DATA
SNDB_GENDER_DATA
SNDB_GROUP
SNDB_GROUP_MESSAGE
SNDB_GROUP_RECIPIENT
SNDB_GROUPS_OF_ROLES
SNDB_LOGGED_IN_DATA
CLOCK_IN_ID
USER_LOGGED_ID
LOGIN_TIME
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
group_user_count
my_first_report
No_Of_Firend_requests
Post_Count
Weekly_Conversation_count
DB_Unimodel3.sql
Meenu_test1.sql
Meenu_test2
Meenu_admin1
mkumbhare
SNDB_Project_Admin~2
SNDB_Project_Admin
SNDB_Project_Admin~1
Meenu_admin1~1
0.597 seconds
Meenu_admin1
Worksheet
Query Builder
650 ON sndb_user_roles_data(lower(Role_name));
651 --drop index SNDB_INDEX_Role_name;
Script Output
Query Result
SQL | 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 --
6 | 0 | SELECT STATEMENT | | 1 | 94 | 1 (0)| 00:00:0
7 | 1 | TABLE ACCESS BY INDEX ROWID BATCHED| SNDB_USER_ROLES_DATA | 1 | 94 | 1 (0)| 00:00:0
8 |* 2 | INDEX RANGE SCAN | SNDB_INDEX_ROLE_NAME | 1 | | 1 (0)| 00:00:0
9 --
10
11 Predicate Information (identified by operation id):
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