IPG

Politécnico
da Guarda

Escola Superior
de Tecnologia e Gestão

Memories

Course:	Data Bases I
Curricular Unit:	None
Academic Year:	2022-2023
Professor: **Student:**	Jose Carlos Fonseca Arazmuhammet Atayev
Date:	04.01.2023

Table of Contents

Memories

Introduction

The Memories application is a platform that allows users to store and share their memories and experiences with others. It is designed to provide a personal and interactive way for users to document and share about their past events, whether they be happy, sad, or otherwise impactful. The application is intended to be used on a variety of devices, including smartphones, tablets, and computers, and will be accessible through a web browser or native app.

Objectives

- To provide users with a platform to create and share memories with friends and family.
- To allow users to organize and store their memories in a user-friendly interface.
- To allow users to search and browse through their memories using a lot of filters and categories.
- To allow users to access their memories from any device with an internet connection.
- To provide a secure platform for storing personal memories and information.

Functionality

The Memories application will allow users to create, view, and edit memories, which will consist of a title, a description, a type (e.g. happy, sad, etc.), a user-defined color, and an optional area or location. Users will be able to search for memories using various filters, such as keyword, type, color, and area. In addition, the application will allow users to share their memories with others through social media platforms or by directly inviting specific users to view their memories.

The infrastructure

The infrastructure required for the memories application to function correctly includes both software and hardware components.
On the software side, the application will require an operating system to run on, such as Windows or Linux. It will also be needed a DB management tool like Oracle. On the hardware side, the application would need a server to host the software and database.

User experience

The memories application will prioritize the user experience by offering a user-friendly interface with intuitive navigation. The design will be responsive, ensuring that the application functions seamlessly across a range of devices. The goal is to provide an enjoyable and easy-to-use experience for the user, encouraging them to continue using the application and adding new memories. To achieve this, the application will be tested on various devices and platforms to ensure compatibility and optimal performance. User feedback will also be regularly collected and used to make any necessary improvements to the interface and overall user experience.

Maintenance and support

The memories application will be designed to be easy to maintain and support. there should be aa dedicated team responsible for addressing any issues that may be, including bugs and other technical problems. There will be also regularly release updates to the application in order to improve its functionality and performance. To provide the best possible support to the users, there will be a number of resources available, such as a detailed FAQ section, a user forum, and a dedicated support team that can be reached through email or phone. Will be made sure to have documentation available to help users troubleshoot any issues they may encounter.

DATA DICTIONARY

NAME	DESCRIPTION	DATA TYPE	SIZE	CONSTRAINTS	KEY	REQUIRED	UNIQUE
SIZE1	THE SIZE OF THE AREA	NUMBER	6,2	PRIMARY KEY, NOT NULL	YES	YES	YES
LAND_NAME	THE NAME OF THE LAND	VARCHAR2	30		NO	NO	NO
COLOR	THE COLOR OF THE AREA	VARCHAR2	30	CHECK CONSTRAINT: IN ('BLACK', 'GREEN', 'PINK', 'YELLOW')	NO	NO	NO
TERRAIN	THE MOOD	VARCHAR2	20	CHECK CONSTRAINT: IN ('HAPPY', 'IMPRESSIVE', 'LONELY', 'SAD', 'TOGETHER')	NO	YES	NO
TYPE	THE TYPE OF AREA	VARCHAR2	20	CHECK CONSTRAINT: IN ('LAND',	NO	NO	NO

				'SEA')			
DEVICE_ID	THE ID OF THE DEVICE	NUMBER	5	PRIMARY KEY, NOT NULL	YES	YES	YES
AVAILIBILTY	WHETHER THE DEVICE IS AVAILABLE OR NOT	CHAR	1	NOT NULL	NO	YES	NO
TYPE	THE TYPE OF DEVICE	VARCHAR2	20	NOT NULL	NO	YES	NO
CAPACITY	THE CAPACITY OF THE DEVICE	VARCHAR2	30	NOT NULL	NO	YES	NO
MEMORY_ID	THE ID OF THE MEMORY	NUMBER	6	PRIMARY KEY, NOT NULL	YES	YES	YES
MEMORY_TITLE	THE TITLE OF THE MEMORY	VARCHAR2	150	NOT NULL	NO	YES	NO
TYPE	THE TYPE OF THE MEMORY	VARCHAR2	20	CHECK CONSTRAINT: IN ('HAPPY', 'IMPRESSIVE', 'LONELY', 'SAD', 'TOGETHER')	NO	YES	NO
USER_USER_ID	THE ID OF THE USER	NUMBER	10	FOREIGN KEY, NOT NULL	NO	YES	YES
COLOR	THE COLOR OF THE MEMORY	VARCHAR2	30	CHECK CONSTRAINT: IN ('BLACK',	NO	YES	NO

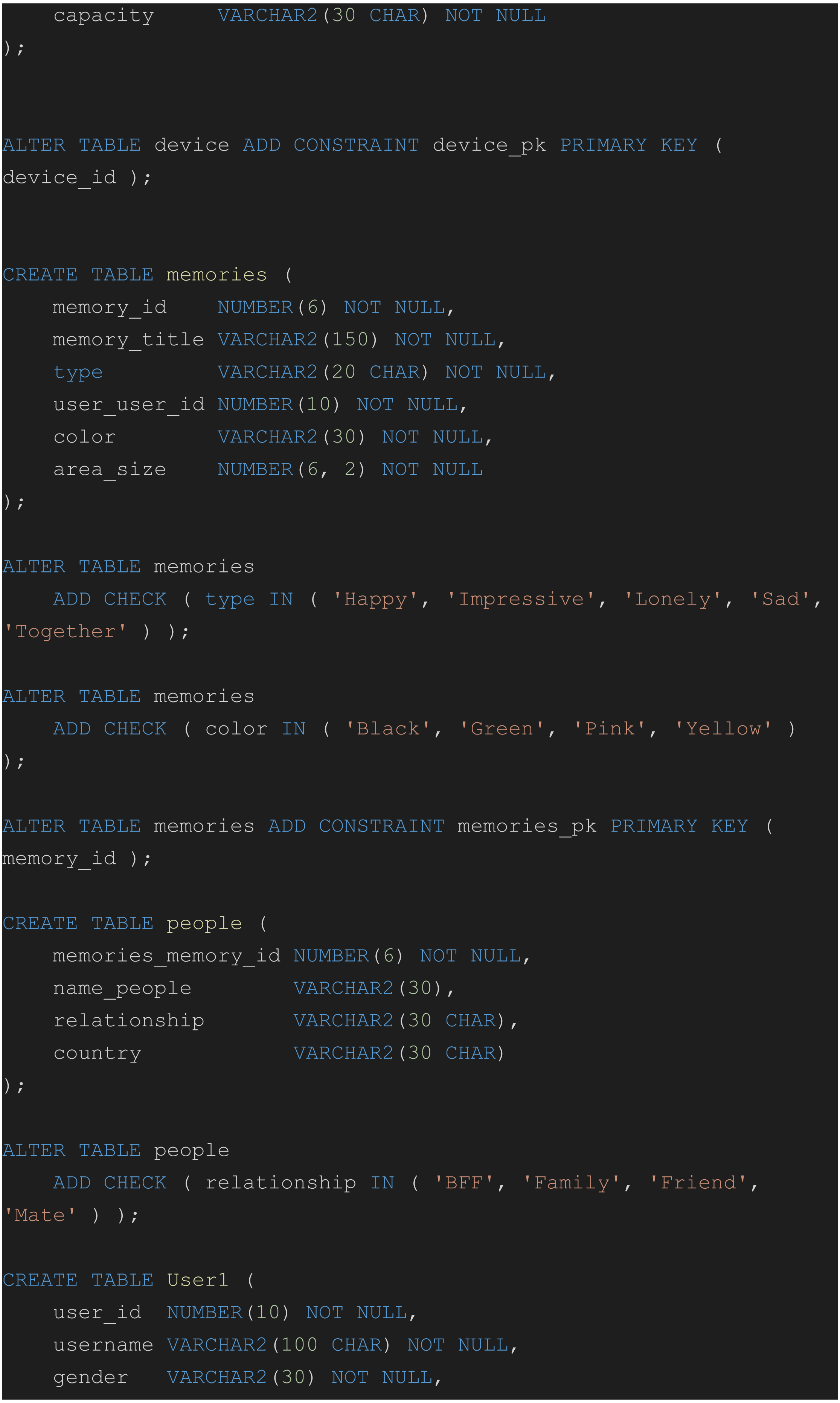
				'GREEN', 'PINK', 'YELLOW')			
AREA_SIZE	THE SIZE OF THE AREA	NUMBER	6,2	FOREIGN KEY, NOT NULL	NO	YES	NO
NAME_PEOPLE	THE NAME OF THE PERSON	VARCHAR2	30	NONE	NO	NO	NO
RELATIONSHIP	THE RELATIONSHIP OF THE PERSON	VARCHAR2	30	CHECK CONSTRAINT: IN ('BFF', 'FAMILY', 'FRIEND', 'MATE')	NO	NO	NO
COUNTRY	THE COUNTRY THE PERSON IS FROM	VARCHAR2	30	NONE	NO	NO	NO
USER_ID	THE ID OF THE USER	NUMBER	10	PRIMARY KEY, NOT NULL	YES	YES	YES
USERNAME	THE USERNAME OF THE USER	VARCHAR2	100	NOT NULL	NO	YES	NO
GENDER	THE GENDER OF THE USER	VARCHAR2	30	CHECK CONSTRAINT : IN (‘FEMALE’, ‘MALE’)	NO	YES	NO
ADRESS	THE ADDRESS OF THE USER	VARCHAR2	250	NONE	NO	NO	NO
AGE	THE AGE OF THE USER	NUMBER	3	NONE	NO	NO	NO

Discussion of Anomalies

1. Update on Memories table - Not affected. Updating the memories table will not affect the people_count column, as it is calculated based on the number of people in the people table.
2. Insert on Memories table - Not affected. Inserting a new row into the memories table will not affect the people_count column
3. Delete on Memories - Not affected. Deleting a row from the memories table will not affect the people_count column
4. Update on People - Not affected. Updating a row in the people table will not affect the people_count column
5. Insert on People - Affected. Inserting a new row into the people table will cause the people_count column in the memories table to be updated, as the trigger created will increment the people_count value by 1.
6. Delete on People - Affected. Deleting a row from the people table will cause the people_count column in the memories table to be updated, as the trigger created will decrement the people_count value by 1.

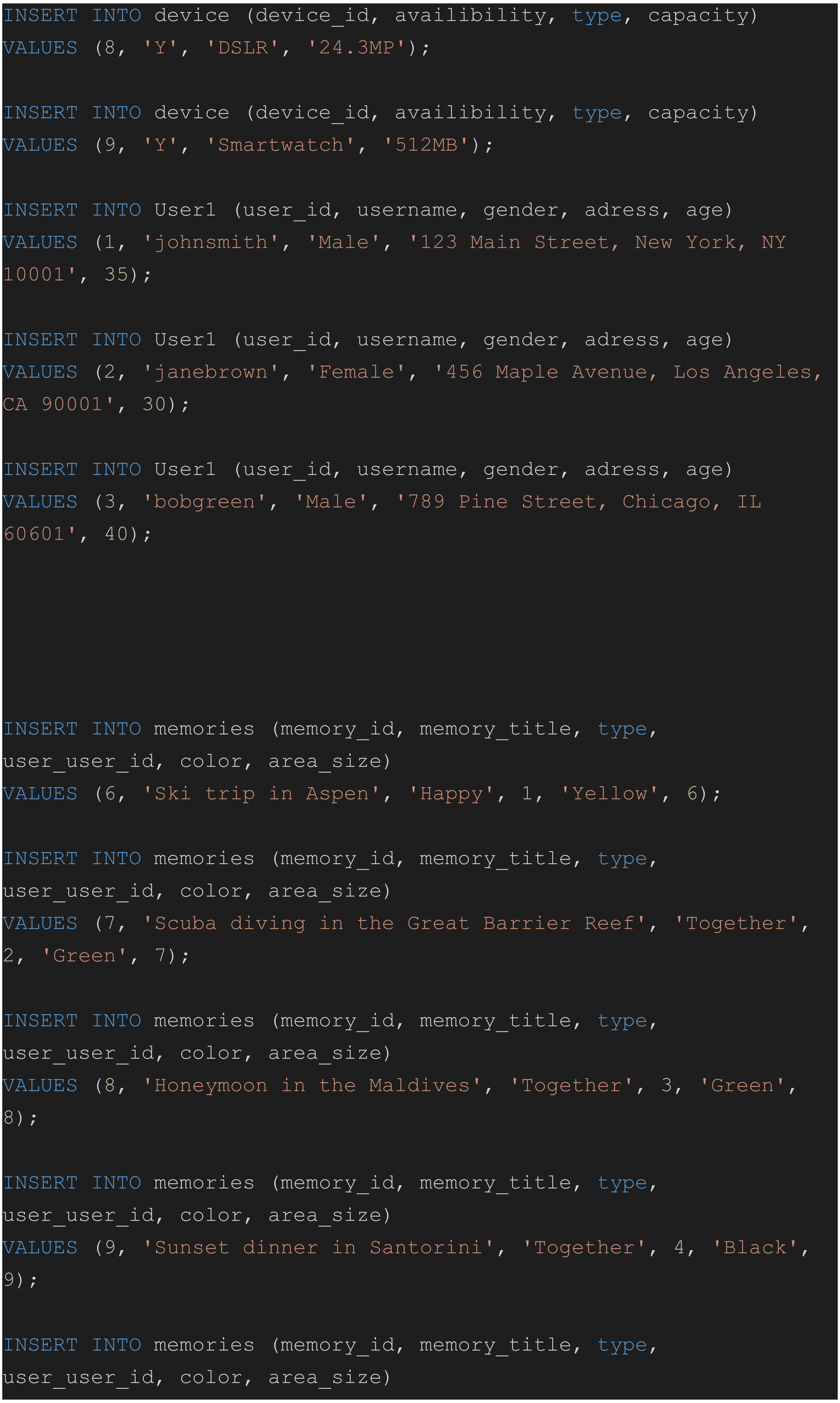
Table Creation & restrictions and Data Insertion

```
-----Creating 6 tables


CREATE TABLE area (
    Size1     NUMBER(6, 2) NOT NULL,
    land_name VARCHAR2(30),
    color     VARCHAR2(30),
    terrain   VARCHAR2(20 CHAR),
    type      VARCHAR2(20 CHAR)
);

ALTER TABLE area
    ADD CHECK ( color IN ( 'Black', 'Green', 'Pink', 'Yellow' )
);

ALTER TABLE area
    ADD CHECK ( terrain IN ( 'Happy', 'Impressive', 'Lonely',
'Sad', 'Together' ) );

ALTER TABLE area
    ADD CHECK ( type IN ( 'Land', 'Sea' ) );

ALTER TABLE area ADD CONSTRAINT area_pk PRIMARY KEY ( Size1 );

CREATE TABLE dev_mem (
    device_device_id    NUMBER(5) NOT NULL,
    memories_memory_id NUMBER(6) NOT NULL
);

ALTER TABLE dev_mem ADD CONSTRAINT relation_6_pk PRIMARY KEY (
device_device_id,

memories_memory_id );

CREATE TABLE device (
    device_id     NUMBER(5) NOT NULL,
    availibility CHAR(1) NOT NULL,
    type          VARCHAR2(20 CHAR) NOT NULL,
```

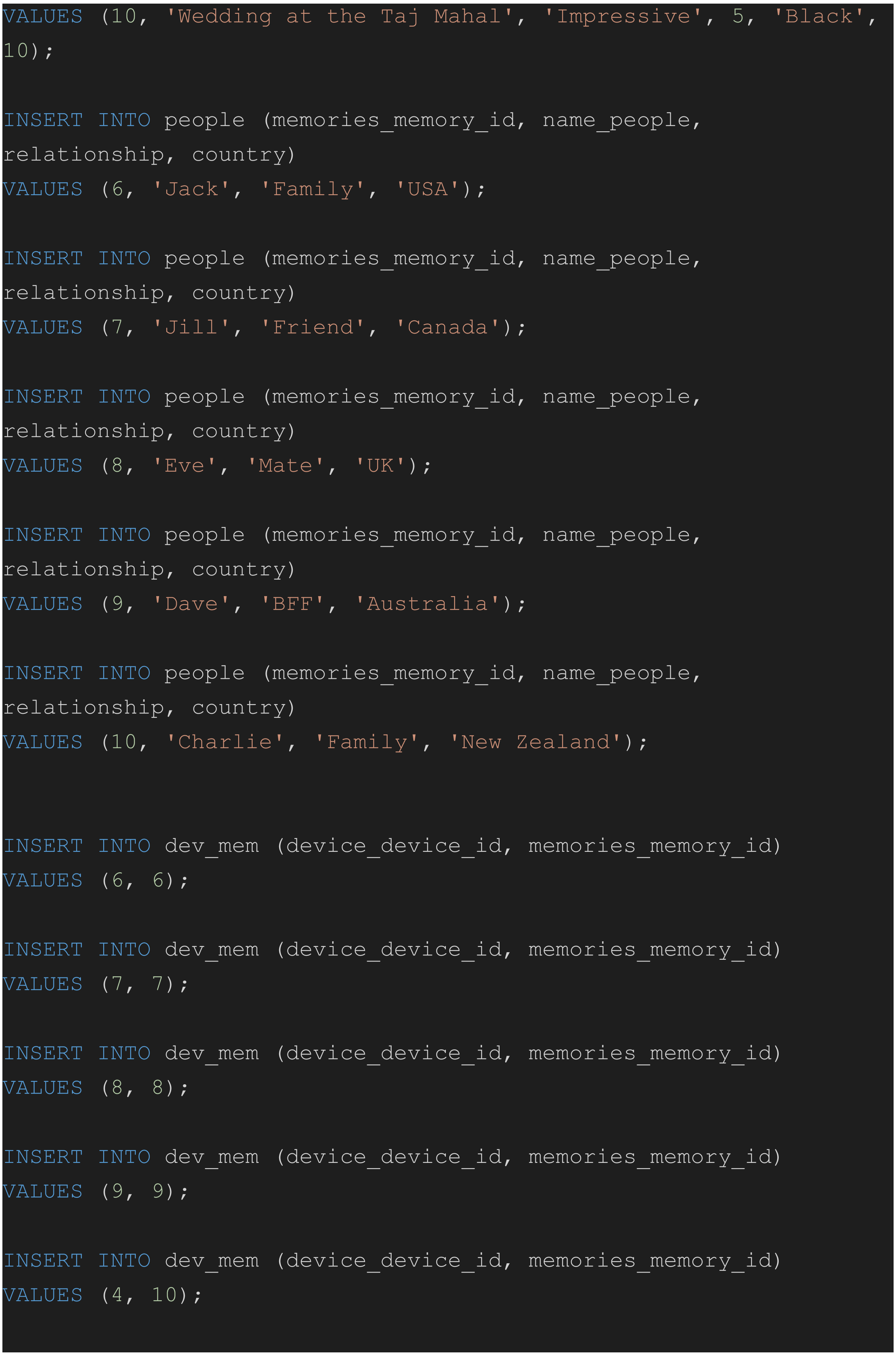


```sql
    capacity     VARCHAR2(30 CHAR) NOT NULL
);


ALTER TABLE device ADD CONSTRAINT device_pk PRIMARY KEY ( 
device_id );


CREATE TABLE memories (
    memory_id    NUMBER(6) NOT NULL,
    memory_title VARCHAR2(150) NOT NULL,
    type         VARCHAR2(20 CHAR) NOT NULL,
    user_user_id NUMBER(10) NOT NULL,
    color        VARCHAR2(30) NOT NULL,
    area_size    NUMBER(6, 2) NOT NULL
);

ALTER TABLE memories
    ADD CHECK ( type IN ( 'Happy', 'Impressive', 'Lonely', 'Sad', 
'Together' ) );

ALTER TABLE memories
    ADD CHECK ( color IN ( 'Black', 'Green', 'Pink', 'Yellow' ) 
);

ALTER TABLE memories ADD CONSTRAINT memories_pk PRIMARY KEY ( 
memory_id );

CREATE TABLE people (
    memories_memory_id NUMBER(6) NOT NULL,
    name_people        VARCHAR2(30),
    relationship       VARCHAR2(30 CHAR),
    country            VARCHAR2(30 CHAR)
);

ALTER TABLE people
    ADD CHECK ( relationship IN ( 'BFF', 'Family', 'Friend', 
'Mate' ) );

CREATE TABLE User1 (
    user_id  NUMBER(10) NOT NULL,
    username VARCHAR2(100 CHAR) NOT NULL,
    gender   VARCHAR2(30) NOT NULL,
    adress    VARCHAR2(250 CHAR),
    age       NUMBER(3)
);

ALTER TABLE User1
    ADD CHECK ( gender IN ( 'Female', 'Male' ) );

ALTER TABLE User1 ADD CONSTRAINT user_pk PRIMARY KEY ( user_id );

ALTER TABLE memories
    ADD CONSTRAINT memories_area_fk FOREIGN KEY ( area_size )
        REFERENCES area ( Size1 );

ALTER TABLE memories
    ADD CONSTRAINT memories_user_fk FOREIGN KEY ( user_user_id )
        REFERENCES User1 ( user_id );

ALTER TABLE people
    ADD CONSTRAINT people_memories_fk FOREIGN KEY (
memories_memory_id )
        REFERENCES memories ( memory_id );

ALTER TABLE dev_mem
    ADD CONSTRAINT relation_6_device_fk FOREIGN KEY (
device_device_id )
        REFERENCES device ( device_id );

ALTER TABLE dev_mem
    ADD CONSTRAINT relation_6_memories_fk FOREIGN KEY (
memories_memory_id )
        REFERENCES memories ( memory_id );







------POPULATING THE DATA


INSERT INTO device (device_id, availibility, type, capacity)
VALUES (4, 'Y', 'Drone', '4K video');
```

```
INSERT INTO User1 (user_id, username, gender, adress, age)
VALUES (4, 'dave123', 'Male', '123 Main St, Kathmandu, Nepal',
29);


INSERT INTO area (Size1, land_name, color, terrain, type)
VALUES (5, 'Serengeti National Park', 'Yellow', 'Together',
'Land');

INSERT INTO device (device_id, availibility, type, capacity)
VALUES (5, 'Y', 'GoPro', 'Full HD video');



INSERT INTO User1 (user_id, username, gender, adress, age)
VALUES (5, 'eve123', 'Female', '123 Main St, Arusha, Tanzania',
32);


INSERT INTO area (Size1, land_name, color, terrain, type)
VALUES (6, 'Yellowstone National Park', 'Yellow', 'Impressive',
'Land');

INSERT INTO area (Size1, land_name, color, terrain, type)
VALUES (7, 'Great Barrier Reef', 'Yellow', 'Together', 'Sea');

INSERT INTO area (Size1, land_name, color, terrain, type)
VALUES (8, 'Maldives', 'Black', 'Together', 'Sea');

INSERT INTO area (Size1, land_name, color, terrain, type)
VALUES (9, 'Santorini', 'Pink', 'Together', 'Land');

INSERT INTO area (Size1, land_name, color, terrain, type)
VALUES (10, 'Taj Mahal', 'Pink', 'Impressive', 'Land');


INSERT INTO device (device_id, availibility, type, capacity)
VALUES (6, 'Y', 'Smartphone', '128GB');

INSERT INTO device (device_id, availibility, type, capacity)
VALUES (7, 'Y', 'GoPro', 'Full HD video');
```



```
INSERT INTO device (device_id, availibility, type, capacity)
VALUES (8, 'Y', 'DSLR', '24.3MP');

INSERT INTO device (device_id, availibility, type, capacity)
VALUES (9, 'Y', 'Smartwatch', '512MB');

INSERT INTO User1 (user_id, username, gender, adress, age)
VALUES (1, 'johnsmith', 'Male', '123 Main Street, New York, NY
10001', 35);

INSERT INTO User1 (user_id, username, gender, adress, age)
VALUES (2, 'janebrown', 'Female', '456 Maple Avenue, Los Angeles,
CA 90001', 30);

INSERT INTO User1 (user_id, username, gender, adress, age)
VALUES (3, 'bobgreen', 'Male', '789 Pine Street, Chicago, IL
60601', 40);





INSERT INTO memories (memory_id, memory_title, type,
user_user_id, color, area_size)
VALUES (6, 'Ski trip in Aspen', 'Happy', 1, 'Yellow', 6);

INSERT INTO memories (memory_id, memory_title, type,
user_user_id, color, area_size)
VALUES (7, 'Scuba diving in the Great Barrier Reef', 'Together',
2, 'Green', 7);

INSERT INTO memories (memory_id, memory_title, type,
user_user_id, color, area_size)
VALUES (8, 'Honeymoon in the Maldives', 'Together', 3, 'Green',
8);

INSERT INTO memories (memory_id, memory_title, type,
user_user_id, color, area_size)
VALUES (9, 'Sunset dinner in Santorini', 'Together', 4, 'Black',
9);

INSERT INTO memories (memory_id, memory_title, type,
user_user_id, color, area_size)
```



```
VALUES (10, 'Wedding at the Taj Mahal', 'Impressive', 5, 'Black',
10);

INSERT INTO people (memories_memory_id, name_people,
relationship, country)
VALUES (6, 'Jack', 'Family', 'USA');

INSERT INTO people (memories_memory_id, name_people,
relationship, country)
VALUES (7, 'Jill', 'Friend', 'Canada');

INSERT INTO people (memories_memory_id, name_people,
relationship, country)
VALUES (8, 'Eve', 'Mate', 'UK');

INSERT INTO people (memories_memory_id, name_people,
relationship, country)
VALUES (9, 'Dave', 'BFF', 'Australia');

INSERT INTO people (memories_memory_id, name_people,
relationship, country)
VALUES (10, 'Charlie', 'Family', 'New Zealand');


INSERT INTO dev_mem (device_device_id, memories_memory_id)
VALUES (6, 6);

INSERT INTO dev_mem (device_device_id, memories_memory_id)
VALUES (7, 7);

INSERT INTO dev_mem (device_device_id, memories_memory_id)
VALUES (8, 8);

INSERT INTO dev_mem (device_device_id, memories_memory_id)
VALUES (9, 9);

INSERT INTO dev_mem (device_device_id, memories_memory_id)
VALUES (4, 10);
```

There are six tables being created in total: "area", "dev_mem", "device", "memories", "people", and "User1".

‘Area’ Table

The "area" table has five columns: "Size1", "land_name", "color", "terrain", and "type". "Size1" is defined as a number data type with a precision of 6 and a scale of 2, and it is set asnot null. The other columns are defined as varchar2 data types with a length of 30 characters for "land_name" and "color", and a length of 20 characters for "terrain" and "type". Check constraints are added to ensure that the values in the "color" and "terrain" columns are from a list of options, and that the value in the "type" column is either "Land" or "Sea". A primary key constraint is also added on the "Size1" column.

‘Dev-Mem’ Table

The "dev_mem" table has two columns: "device_device_id" and "memories_memory_id". Both are defined as number data types with a precision of 5 and 6, respectively. A primary key constraint is added on both columns.

‘Device’ Table

The "device" table has four columns: "device_id", "availibility", "type", and "capacity". "device_id" is defined as a number data type with a precision of 5, and "availibility" is defined as a char data type with a length of 1. "type" and "capacity" are both defined as varchar2 data types with a length of 20 and 30 characters, respectively. A primary key constraint is added on the "device_id" column.

‘Memories’ Table

The "memories" table has six columns: "memory_id", "memory_title", "type", "user_user_id", "color", and "area_size". "memory_id" is defined as a number data type with a precision of 6, and "user_user_id" is defined as a number data type with a precision of 10. "memory_title" and "color" are defined as varchar2 data types with a length of 150 and 30 characters, respectively. "type" is defined as a varchar2 data type with a length of 20 characters, and a Check constraint is added to ensure that the values are from a predetermined list of options. "color" also has a Check constraint to ensure that the values are from a predetermined list of options. A primary key constraint is added on the "memory_id" column.

‘People’ table

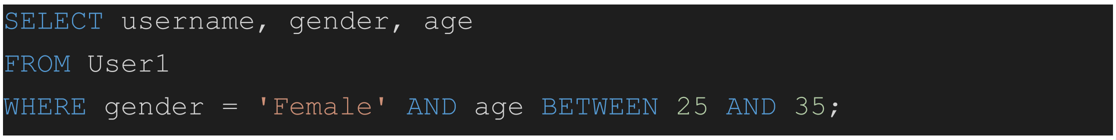
The "people" table has four columns: "memories_memory_id", "name_people", "relationship", and "country". "memories_memory_id" is defined as a numberr data type with a precision of 6. "name_people" is defined as a varhcar2 data type with a length of 30 characters. "relationship" is defined as a varchar2 data type with a length of 30 characters, and a cechk constraint is added to ensure that the values are from a predetermined list of options. "country" is defined as a varchar2 data type with a length of 30 characters.

‘User1’ Table

The "User1" table has five columns: "user_id", "username", "gender", "adress", and "age". "user_id" is defined as a number data type with a precision of 10. "username" is defined as a varchar2 data type with a length of 100 characters. "gender" is defned as a varcahr2 data type with a length of 30 characters, and a check constraint is added to ensure that the values are either "Female" or "Male". "adress" is defined as a varchar2 data type with a length of 250 characters. "age" is defined as a numberr data type with a precision of 3. A primary key constraint is added on the "user_id" column.

SQL (Complexity)

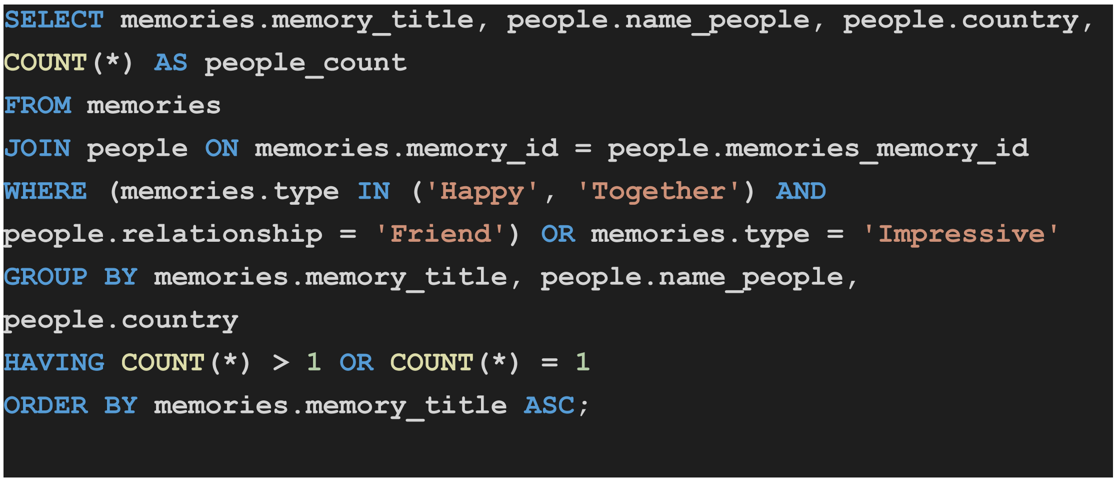
Code #1



```sql
SELECT username, gender, age
FROM User1
WHERE gender = 'Female' AND age BETWEEN 25 AND 35;
```

This query will select the username, gender, and age columns from the User1 table for all rows where the gender is 'Female' and the age is between 25 and 35 (inclusive).

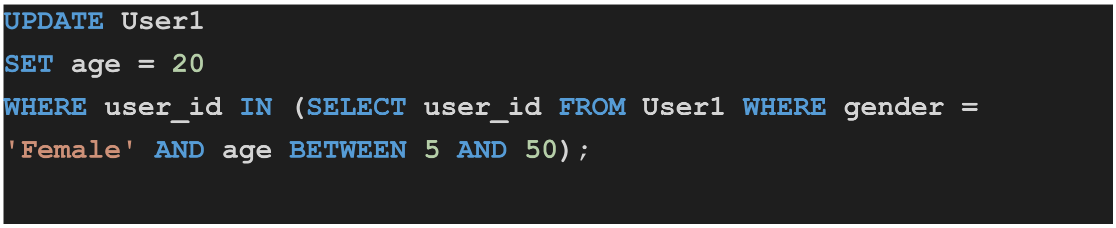
Code #2



```sql
SELECT memories.memory_title, people.name_people, people.country,
COUNT(*) AS people_count
FROM memories
JOIN people ON memories.memory_id = people.memories_memory_id
WHERE (memories.type IN ('Happy', 'Together') AND
people.relationship = 'Friend') OR memories.type = 'Impressive'
GROUP BY memories.memory_title, people.name_people,
people.country
HAVING COUNT(*) > 1 OR COUNT(*) = 1
ORDER BY memories.memory_title ASC;
```

This selects the memory_title column from the memories table and the name_people and country columns from the people table. It will then return all rows where the type column in the memories table is either 'Happy' or 'Together', and the relationship column in the people table is 'Friend' orr the type column in the memories table is 'Impressive'. then it will group the results by the memory_title, name_people, and country columns and use the Having clause to only return rows where the count of the grouped results is greater than 1 or equal to one. In tje end the results are ordered by the memory_title column in increasing order.

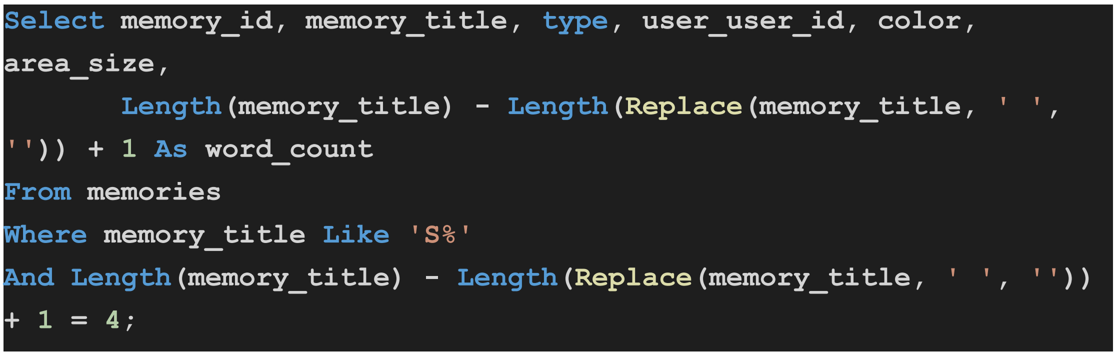
Code #3



```
UPDATE User1
SET age = 20
WHERE user_id IN (SELECT user_id FROM User1 WHERE gender =
'Female' AND age BETWEEN 5 AND 50);
```

It first compiles the Select subquery that there it filters all females who are in age between 5 and 50 and after that it updates all of their ages to 20 years old.

Code #4

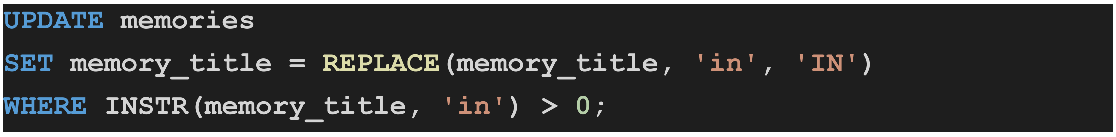


```
Select memory_id, memory_title, type, user_user_id, color,
area_size,
       Length(memory_title) - Length(Replace(memory_title, ' ',
'')) + 1 As word_count
From memories
Where memory_title Like 'S%'
And Length(memory_title) - Length(Replace(memory_title, ' ', ''))
+ 1 = 4;
```

Here it selects the memory ID, memory title, type, user ID, color, area size, and the word count of each memory from the "memories" table.

The word count column is calculated by subtracting the length of the memory title string with the length of the same string with all spaces removed, then adding 1. The resulting rows are filtered to only show memories with a memory title that starts with "S" and has 4 words.

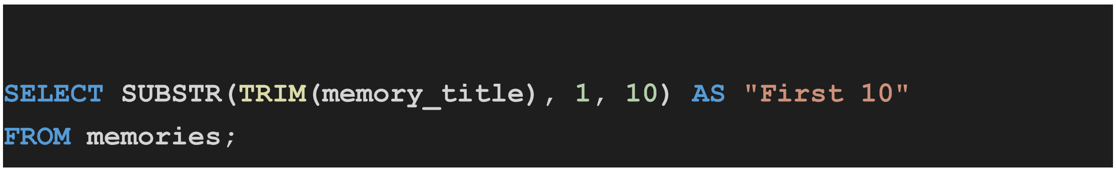
Code #5



```
UPDATE memories
SET memory_title = REPLACE(memory_title, 'in', 'IN')
WHERE INSTR(memory_title, 'in') > 0;
```

It updates the memory_title column in the memories table by replacing all instances of the substring 'in' with 'IN', as long as 'in' appears in the memory_title string.

Code #6

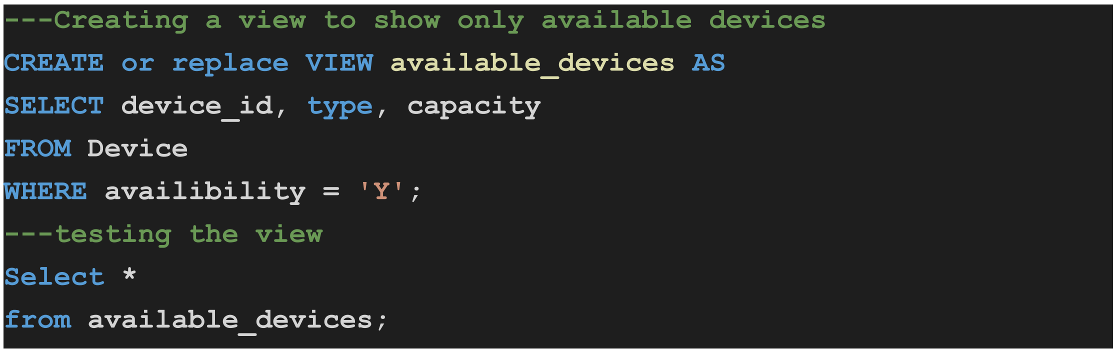


```
SELECT SUBSTR(TRIM(memory_title), 1, 10) AS "First 10"
FROM memories;
```

Selects the first 10 characters of the memory_title field after trimming any leading or trailing whitespace. The resulting value will displau as column First ten.

Views, Sequences and Synonyms

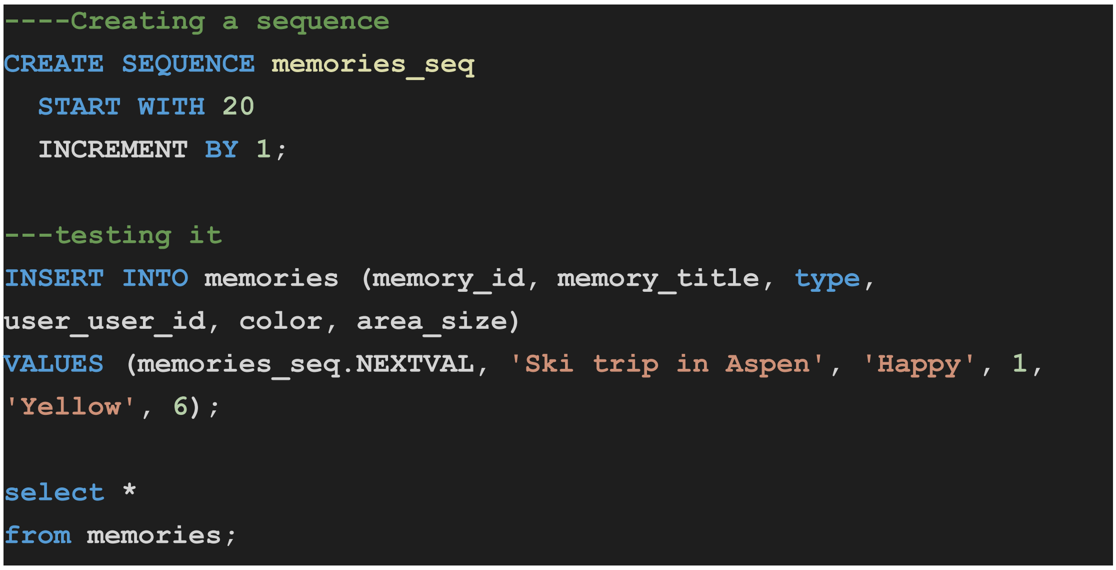
Code #7



```
---Creating a view to show only available devices
CREATE or replace VIEW available_devices AS
SELECT device_id, type, capacity
FROM Device
WHERE availibility = 'Y';
---testing the view
Select *
from available_devices;
```

This code creates a view called **available_devices** that displays the device ID, type, and capacity of all devices that have an availibility of 'Y' - Yes..

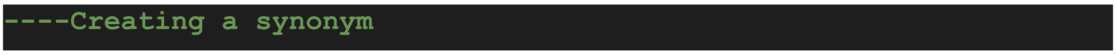
Code #8



```
----Creating a sequence
CREATE SEQUENCE memories_seq
  START WITH 20
  INCREMENT BY 1;

---testing it
INSERT INTO memories (memory_id, memory_title, type,
user_user_id, color, area_size)
VALUES (memories_seq.NEXTVAL, 'Ski trip in Aspen', 'Happy', 1,
'Yellow', 6);

select *
from memories;
```

Here, it first creates a sequence called memories_seq which starts with 20 and with increment of 1. then when we add a memory, memory_ID will be 20, 21, 22, 23 like that order. We can see the usage the sequences here.

Code #9



```
----Creating a synonym
```

```
CREATE SYNONYM people_syn FOR people;

--testing it
SELECT * FROM people_syn
WHERE (relationship = 'BFF' AND country = 'USA')
OR (relationship = 'Family' AND country = 'Nepal');
```

This creates a synonym called **people_syn** for the people table. synonym "people_syn" can be used in place of the table name people.

By using select the synonym **people_syn** selects all rows from the people table where the relationship column is equal to 'BFF' and the country column is equal to 'USA'. The result set includes the memories_memory_id, name_people, and relationship columns from the people table for the rows that meet the criteria.

Privileges and Roles

Code #10

```
---Creating a role for administrators
CREATE ROLE administrator_role;
GRANT CREATE, SELECT, INSERT, UPDATE,
DELETE ON ALL TABLES TO administrator_role;

-----Creating a role for users
CREATE ROLE users;
GRANT INSERT, SELECT, UPDATE, DELETE ON user1 TO users;
GRANT INSERT, SELECT, UPDATE, DELETE ON memories TO users;
GRANT SELECT ON people TO users;
```

First, a role has been created for administrators, administrator_role which has access to all the tables. And then Users role has been created for users which has specific accesses. You can see it in the CRUD table below:

	Device	**Dev_mem**	**Memories**	**People**	**User1**	**Area**
Admin	CRUD	CRUD	CRUD	CRUD	CRUD	CRUD
User			CRUD	R	CRUD	

Table 1: CRUD Table

Transactions

Code #11

```
BEGIN
    INSERT INTO people (memories_memory_id, name_people,
relationship, country)
    VALUES (100, 'John', 'Friend', 'USA');

    UPDATE memories
    SET people_count = people_count + 1
    WHERE memory_id = 100;

    -- commit the transaction if there s no error
    COMMIT;
EXCEPTION
    -- If any error, then roll back the transaction
    WHEN OTHERS THEN
        ROLLBACK;
END;
```

It tries to insert a row into people table and after that increase people_count by 1 and it commits it. But if there is some kind of error/exception, then rollback.

PL/SQL

Code #12

```
--creating a procedure, cursor, loops, iteration
CREATE OR REPLACE PROCEDURE iterate_user1_rows (num_rows IN
INTEGER)
AS
  CURSOR user1_cur IS
    SELECT * FROM User1;
  user1_rec user1_cur%ROWTYPE;
BEGIN
  FOR i IN 1..num_rows
  LOOP
    FETCH user1_cur INTO user1_rec;
    DBMS_OUTPUT.PUT_LINE(user1_rec.user_id || ' - ' ||
user1_rec.username || ' - ' || user1_rec.gender || ' - ' ||
user1_rec.adress || ' - ' || user1_rec.age);
  END LOOP;
  CLOSE user1_cur;
END;
/


----to test
BEGIN
  iterate_user1_rows(5);
END;
/
```

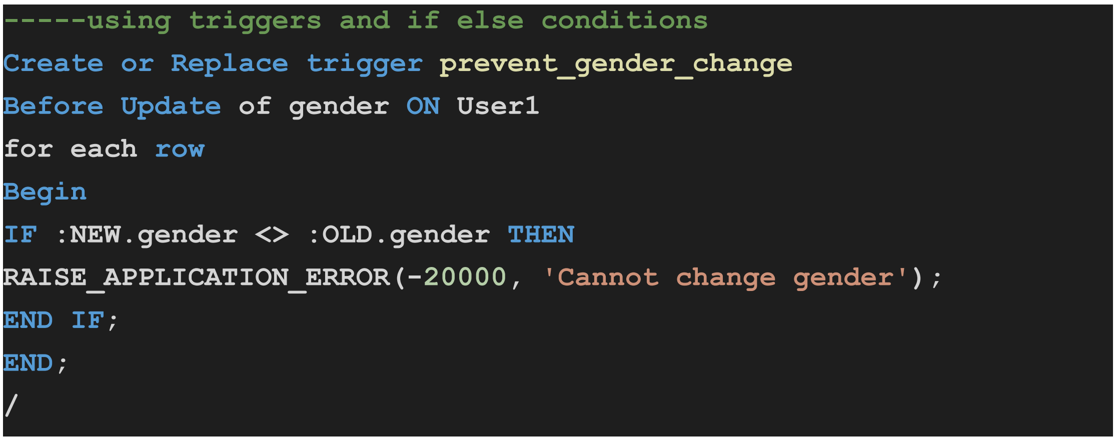
This procedure iterate_user1_rows accepts a number as a parameter, an num_rows. A cursor user1_cur iscreated, which iterates through all rows of the User1 table. A record called user1_rec is declared, using the %rowtype attribute of the cursor to define. then enters a loop that iterates the number of times, specified by the num_rows input parameter. On each iteration, it take the next row from the cursor into the "user1_rec" record and using the DBMS it outputs the information. after the loop finishes, the cursor is removed/closed. And in the end we are testing it by giving a paramtetr of 5 to iterate_user1_rows(5).

Code #13

```
----Creating the function
CREATE OR REPLACE FUNCTION get_memory_title (p_input_string IN
VARCHAR2)
RETURN VARCHAR2
AS
  l_output_string VARCHAR2(255);
BEGIN
  SELECT memory_title INTO l_output_string
  FROM memories
  WHERE memory_title = p_input_string;

  RETURN l_output_string;
END;
/

-----testing the function
BEGIN
  DBMS_OUTPUT.PUT_LINE(get_memory_title('Honeymoon IN the
Maldives'));
END;
/
```

This function takes a string input, which is defined as p_input_string, and returns a string output because it is a function. The function is defined with the create or replace statement, which creates the function or replaces it if it already exists. Inside the function, a local variable l_output_string is defined as a string with a maximum length of 255 characters. This variable will be used to store the memory title that is retrieved from the memories table. The function then uses a select statement to get the memory_title from the memories table, on the condition that the memory_title must match the value of p_input_string. The result of this select statement is stored in the variable l_output_string. And the return statement return the l_output_string . Then in order to test, we are calling it with DBMS output function/option.

Code #14



```
-----using triggers and if else conditions
Create or Replace trigger prevent_gender_change
Before Update of gender ON User1
for each row
Begin
IF :NEW.gender <> :OLD.gender THEN
RAISE_APPLICATION_ERROR(-20000, 'Cannot change gender');
END IF;
END;
/
```

This trigger is made to not to allow users from updating the gender column in the User1 table. When an update operation is performed on the User1 table, the trigger will execute and check the new value for the gender column. If the new value is different from the old value, the trigger will raise an exception and the update will not be allowed to complete. If the new value is the same as the old value, the update will be allowed to proceed as normal.

BIBLIOGRAPHY

- https://www.ibm.com/docs/en/db2/11.5?topic=plsql-using-rowtype-cursors
- https://docs.oracle.com/database/121/SQLRF/statements_7002.htm
- Oracle Procedure - javatpoint