Q1:

Load the data in the given Excel csv file to a database table as-is using SQL Developer. Show all steps and the loaded table.

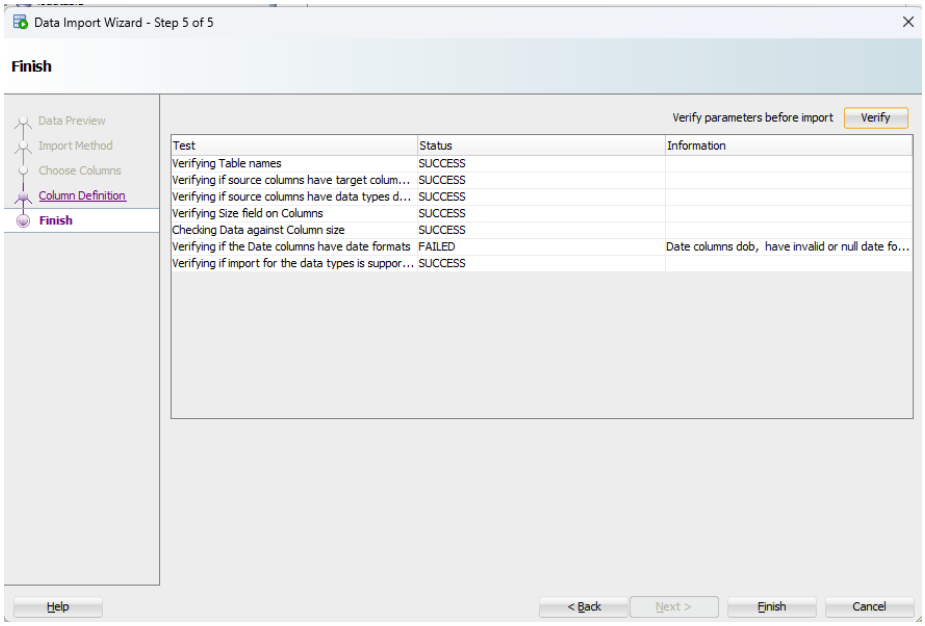

Figure 1: tableload_success

Q2:

From the table in Q1, using SQL create the student table (student_id, name varchar, dob) Create the keys using a student table sequence. Student name and dob cannot be NULL. Test the constraints using SQL and show the results. You have to test the primary key constraint, and the null constraint on the 2 columns.

```
-- create student table
-- number, varchar, date
-- clear table
DROP TABLE STUDENT;

-- Create student talbe
CREATE TABLE student (
    student_id NUMBER PRIMARY KEY,
    name VARCHAR2(20) NOT NULL,
    dob DATE NOT NULL
```

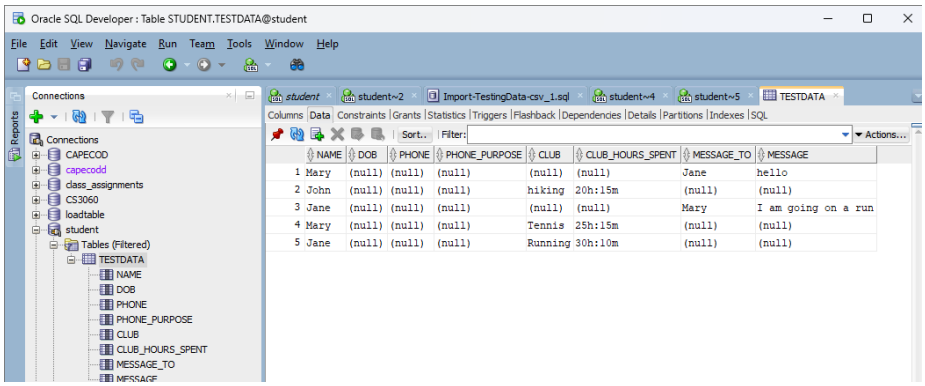

Figure 2: testdataTable

```
);

-- Drop the sequence
DROP SEQUENCE student_id_seq;

-- create sequence for student ID
CREATE SEQUENCE student_id_seq
START WITH 1
INCREMENT BY 1;

-- load table with student information
-- Mary
INSERT INTO STUDENT (
    STUDENT_ID,
    NAME,
    DOB)

VALUES (student_id_seq.nextval, 'Mary', '03-SEP-2024');

-- update new student Jane
INSERT INTO STUDENT (
    STUDENT_ID,
    NAME,
    DOB)

VALUES (student_id_seq.nextval, 'Jane', '01-SEP-2024');

select * from student;

-- test null values
```

```
INSERT INTO STUDENT (
    STUDENT_ID,
    NAME,
    DOB)
VALUES ('', '', '');

-- Show table
select * from student;

-- test primary key values
INSERT INTO STUDENT (
    STUDENT_ID,
    NAME,
    DOB)
VALUES (1, 'Michael', '30-AUG-2024');

-- Show table
select * from student;
```

Q2 Output

> table STUDENT_HISTORY dropped. table STUDENT_HISTORY created. TRIGGER STUDENT_UPDATE_HISTORY compiled 1 rows updated. OLD_STUDENT_ID OLD_NAME TIME_STAMP ———— ———— ——————————- 1 Bob Saget 06-SEP-24 05.45.44.226000000 PM

Q3:

> From the table in Q1 create a club table (club_id, club_name)
>
> Club examples are hiking, running and tennis. Club_id must be generated using a club sequence and the club-name cannot be null. Test the constraints using SQL and show the results. You have to test the primary key and constraint, and the null constraint on 1 column.

```
DROP TABLE CLUB;
-- create club table
CREATE TABLE CLUB (
    club_id NUMBER PRIMARY KEY,
    club_name VARCHAR2(20) not null
);

-- drop sequence
drop sequence club_id_seq;
```

```sql
-- generate sequence for club ID
CREATE SEQUENCE club_id_seq
START WITH 1
INCREMENT BY 1;

-- insert new rows for hiking, running and tennis
INSERT INTO CLUB (club_id, club_name)
VALUES (club_id_seq.nextval, 'HIKING');

INSERT INTO CLUB (club_id, club_name)
VALUES (club_id_seq.nextval, 'RUNNING');

INSERT INTO CLUB (club_id, club_name)
VALUES (club_id_seq.nextval, 'TENNIS');

-- test primary key and "NULL" values
INSERT INTO CLUB (club_id, club_name)
VALUES (1,'HIKING');

-- check output
select * from club;
insert into club (club_id, club_name) values (5, '');

-- check output
select * from club;
```

Q3 Output

table CLUB dropped. table CLUB created. sequence CLUB_ID_SEQ dropped. sequence CLUB_ID_SEQ created. 1 rows inserted. 1 rows inserted. 1 rows inserted.

Error starting at line : 27 in command - INSERT INTO CLUB (club_id, club_name) VALUES (1,'HIKING') Error report - SQL Error: ORA-00001: unique constraint (STUDENT.SYS_C007307) violated 00001. 00000 - "unique constraint (%s.%s) violated" *Cause: An UPDATE or INSERT statement attempted to insert a duplicate key. For Trusted Oracle configured in DBMS MAC mode, you may see this message if a duplicate entry exists at a different level.* Action: Either remove the unique restriction or do not insert the key. CLUB_ID CLUB_NAME ———- ————— 1 HIKING 2 RUNNING 3 TENNIS

Error starting at line : 32 in command - insert into club (club_id, club_name) values (5, '') Error report - SQL Error: ORA-01400: cannot insert NULL into ("STUDENT"."CLUB"."CLUB_NAME") 01400.

00000 - “cannot insert NULL into (%s)” *Cause:* Action: CLUB_ID CLUB_NAME ———- ————— 1 HIKING 2 RUNNING 3 TENNIS

Q4:

From the table in Q1 create a messages table (message_from, message_to, message) Example message is from Mary to Jane and Hello. message_from & message_to should be foreign keys to the student table, and the message cannot be null. message_from & message_to are student_ids. Test the constraints using SQL and show the results. You have to test the foreign key and constraint, and the null constraint on 1 column.

```
-- Q4
-- create messages table
drop table messages;

-- messages_from and message_to foreign keys in the student table
create table messages (
    message_id number primary key,
    message_from number not null,
    message_to number not null,
    message varchar2(50) not null,  -- messages cannot be 'NULL'

-- message from and to are student_ids
foreign key (message_from) references student(student_id),
foreign key (message_to) references student(student_id));

-- Create a sequence for message_id
create sequence message_id_seq
start with 1
increment by 1;

-- Test message Mary to Jane is "hello"
insert into messages (message_id, message_from, message_to, message)
values (message_id_seq.nextval,
        (select student_id from STUDENT where name = 'Mary'),
        (select student_id from STUDENT where name = 'Jane'),
        'Hello');

-- show output
select * from messages;

-- Test foreign key constraint
```

```
-- (non-existent student_id 999 for message_from)
insert into messages (message_id, message_from, message_to, message)
values (message_id_seq.nextval,
        (select student_id from STUDENT where student_id = 999),
        (select student_id from STUDENT where student_id = 1),
        'Hello');

-- Test NULL constraint
insert into messages (message_id, message_from, message_to, message)
values (message_id_seq.nextval,
        (select student_id from STUDENT where name = 'Mary'),
        (select student_id from STUDENT where name = 'Jane'),
        NULL);

-- show output
select * from messages;

-- create a table called club_time_spent
drop table phone;

-- cols are student_id and time_in_minutes
create table phone (

student_id number primary key,
time number,

foreign key (student_id) references student(student_id));
```

> You will also need to create a table called Club_Time_Spent (student_id, time_in_minutes). This table also should have the foreign key constraints and the null constraints. You will need to do data wrangling to store the time in minutes. You will use this table in an SQL later.

```
-- Q4 cont.
-- create a table called club_time_spent
drop table club_time_spent;

-- cols are student_id and time_in_minutes
create table club_time_spent(
    time_in_minutes time primary key,
    student_id number,
foreign key (student_id) references student(student_id));
```

Club Table

```
table CLUB_ACTIVITY_LOG dropped.
table CLUB_ACTIVITY_LOG created.
1 rows inserted.
1 rows inserted.
```

STUDENT_ID	TIME_IN_MINUTES
	90
2	60

Q5:

From the table in Q1 create the phone table (student_id, phone, phone_purpose) Phone_purpose can only be cell, work or home. Test the constraints using SQL and show the results. You have to test the foreign key constraint, and the constraint on the phone_purpose column.

```sql
--Q5
--student id, phone_num, phone_purpose(
-- purpose is either cell, work, or home

drop table phone_purpose_table;
create table phone_purpose_table (
phone_type char(4) primary key);

-- Insert allowed phone purposes into phone_purpose_table
insert into phone_purpose_table (phone_type) values ('cell');
insert into phone_purpose_table (phone_type) values ('work');
insert into phone_purpose_table (phone_type) values ('home');

drop table phone_table;
-- add phone table
create table phone_table (
    student_id number,
    phone varchar2(10) not null,
    phone_purpose char(4) not null,
    primary key (student_id, phone_purpose),
    foreign key (phone_purpose) references phone_purpose_table(phone_type),
    foreign key (student_id) references student(student_id)
);

-- insert new entry for mary and her phone
insert into phone_table (student_id, phone, phone_purpose)
values ((select student_id from student where name = 'Mary'),'1234567890','cell');
```

```sql
insert into phone_table (student_id, phone, phone_purpose)
values ((select student_id from student where name = 'Mary'),'0987654321','work');

select * from phone_table;

insert into phone_table (student_id, phone, phone_purpose)
values ((select student_id from student where name = 'Mary'),'0987654321','mobl');

select * from phone_table;
```

Q6:

For the Student table create a history table that stores the old student row with timestamp (Sysdate in Oracle) on update of student row, using a PL/SQL procedure. Test this by updating a student row and thus creating an entry in the student-history table. Show the before and after of the tables.

```sql
drop table student_history;

-- create history table with the old student row w/ timestamp
create table student_history (
    old_student_id number,
    old_name varchar2(20),
    time_stamp timestamp default systimestamp, -- sysdate or systimestamp
    foreign key (old_student_id) references student(student_id)
);
-- test by creating update in student row, should generate student_history table row as well
create or replace trigger student_update_history
before update on student
for each row
begin
insert into student_history (old_student_id, old_name, time_stamp)
values ( :old.student_id, :old.name, systimestamp );
end;
/
--test trigger
update student
set name = 'Bob Saget'
where student_id = 1;

select * from student_history;
```

SQL Output

table STUDENT_HISTORY dropped. table STUDENT_HISTORY

created. TRIGGER STUDENT_UPDATE_HISTORY compiled 1 rows updated.

OLD_STUDENT_ID	OLD_NAME	TIME_STAMP
1	Bob Saget	06-SEP-24 05.45.44.226000000 PM

Q7:

:> Create a View that shows the message-from (student), the message-to (student), their dobs, their phone and the message sent, and order by dob of message-from student. dob is a date column and not a string. Do a select from the View to show all the rows. dob should show as MM-DD-YYYY and phone-number should show in the format XXX-XXX-XXXX.

```
-- create view
/*
View must have
FROM -> TO
order by DOB of from
date is a col not string
DOB is date format MM-DD-YYYY

phone nubmer is XXX-XXX-XXXX
*/

-- `CREATE OR REPLACE` **updates or modifies** an existing view without needing to first `dr

create or replace student_message_view as
select
    student_message_table.message_from,
    student_message_table.message_to,
    student_message_table.message,
from student_message_table;


-- This view selects the name from the student table where the student_id matches message_fr

create or replace view student_message_view as
select
    (select name from student where student.student_id = student_message_table.message_from)
    (select name from student where student.student_id = student_message_table.message_to) a
    student_message_table.message
from student_message_table;
```

```
-- adding dob to the view.

create or replace view student_message_view as
select

    (select name from student where student.student_id = student_message_table.message_from)
    (select dob from student where student.id = student_message_table.message_from) as sende
    (select phone_number from student where student.student_id = student_message_table.messa


    (select name from student where student.student_id = student_message_table.message_to) a
    (select dob from student where student.id = student_message_table.message_to) as recieve
    (select phone_number from student where student.student_id = student_message_table.messa

    student_message_table.message
from student_message_table;


/*CREATE VIEW student_message_view AS
SELECT
FROM student_message_table as from

WHERE condition;

select * from student message view
*/
```

Q8:

Using a Java Metadata program, show the metadata for the Student and Club tables only. There is no need to show the DB metadata, just the information for the 2 tables.

```
import java.sql.*;

public class TestDBMetaData {

    public static void main (String[] args) {

        try {
            Class.forName("oracle.jdbc.OracleDriver");
            System.out.println("Driver loaded");

            String url = "jdbc:oracle:thin:@localhost:1521:xe";
```

```
            String user = "student";
            String pwd = "win11";

            Connection DB_mobile_conn = DriverManager.getConnection(url,user,pwd);
            System.out.println("Database Connect ok");
            System.out.println(" ");

            DatabaseMetaData dmd = DB_mobile_conn.getMetaData();

            ResultSet studentColumns = dmd.getColumns(null, null,"student", null);
            System.out.println("Metadata Student Table");

            while (studentColumns.next()) {
            System.out.println(" " + studentColumns.getString("Column Name") + " " +  studer
            }

            studentColumns.close();


            while (studentColumns.next()) {
            System.out.println(" " + studentColumns.getString("Column Name") + " " +  studer
            }
            clubColumns.close();
            DB_mobile_conn.close();
        } catch (Exception exp) {
            System.out.println("Exception = " +exp)
        }
    }
}
```

```
PS C:\dev\javaDev> java -cp ".;C:\dev\javaDev\ojdbc7.jar" q8MetaData1
Driver loaded
Database Connect ok

Metadata for Student Table:
 STUDENT_ID NUMBER 0
 NAME VARCHAR2 20
 DOB DATE 7
 STUDENTID NUMBER 0
 STUDENTNICK VARCHAR2 50
Metadata for Club Table:
 CLUB_ID NUMBER 0
 CLUB_NAME VARCHAR2 20
PS C:\dev\javaDev>
```

Q9:

Using a Java program SQL inject the Student table and using a Java Prepared statement show that the SQL injection can be prevented. Show your work by running the prograqm and output.

```
import java.sql.*;
import java.util.Scanner;

public class SQLInjectionExample {

    public static void main(String[] args) {

        try {
            // Load Oracle JDBC driver
            Class.forName("oracle.jdbc.OracleDriver");
            System.out.println("Driver loaded");

            // Establish a connection
            String url = "jdbc:oracle:thin:@localhost:1521:xe";
            String user = "student";
            String pwd = "win11";
            Connection connection = DriverManager.getConnection(url, user, pwd);
            System.out.println("Database Connect ok");

            // Take input from the user (simulate the injection)
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter Student ID: ");
            String studentId = scanner.nextLine();

            // Vulnerable query (concatenating user input directly into the query)
            String sql = "SELECT * FROM STUDENT WHERE STUDENT_ID = '" + studentId + "'";
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);

            // Show the query results (or a failure message)
            if (resultSet.next()) {
                System.out.println("Student ID: " + resultSet.getString("STUDENT_ID"));
                System.out.println("Name: " + resultSet.getString("NAME"));
                System.out.println("DOB: " + resultSet.getDate("DOB"));
            } else {
                System.out.println("No student found with the given ID.");
            }

            // Close resources
            resultSet.close();
```

```
            statement.close();
            connection.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
SELECT * FROM STUDENT WHERE STUDENT_ID = '' OR '1'='1'
```

Q10.

Write an SQL query that shows all students, their dob, their cell phone numbers, the clubs they are members of and the total time they spent in the club in minutes. Test it in SQL developer, and then run it in Java and show the ResultSet metadata.

```
SELECT
    STUDENT.STUDENT_ID,
    STUDENT.NAME,
    STUDENT.DOB,
    STUDENT.CELL_PHONE_NUMBER,
    CLUB.CLUB_NAME,
    CLUB_MEMBERSHIP.TIME_SPENT_IN_MINUTES
FROM
    STUDENT
JOIN
    CLUB_MEMBERSHIP ON STUDENT.STUDENT_ID = CLUB_MEMBERSHIP.STUDENT_ID
JOIN
    CLUB ON CLUB_MEMBERSHIP.CLUB_ID = CLUB.CLUB_ID;
```

```
import java.sql.*;

public class ClubMembershipQuery {

    public static void main(String[] args) {

        try {
            // Load Oracle JDBC driver
            Class.forName("oracle.jdbc.OracleDriver");
            System.out.println("Driver loaded");

            // Establish a connection
            String url = "jdbc:oracle:thin:@localhost:1521:xe";
            String user = "student";
```

```
            String pwd = "win11";
            Connection connection = DriverManager.getConnection(url, user, pwd);
            System.out.println("Database Connect ok");

            // SQL Query to fetch students, DOB, phone numbers, club memberships, and total
            String query = "SELECT STUDENT.STUDENT_ID, STUDENT.NAME, STUDENT.DOB, STUDENT.CE
                           "CLUB.CLUB_NAME, CLUB_MEMBERSHIP.TIME_SPENT_IN_MINUTES " +
                           "FROM STUDENT " +
                           "JOIN CLUB_MEMBERSHIP ON STUDENT.STUDENT_ID = CLUB_MEMBERSHIP.STU
                           "JOIN CLUB ON CLUB_MEMBERSHIP.CLUB_ID = CLUB.CLUB_ID";

            // Create a Statement object and execute the query
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);

            // Get metadata from the result set
            ResultSetMetaData rsMetaData = resultSet.getMetaData();
            int columnCount = rsMetaData.getColumnCount();

            // Display ResultSet metadata
            System.out.println("ResultSet Metadata:");
            for (int i = 1; i <= columnCount; i++) {
                System.out.println("Column " + i + ": " + rsMetaData.getColumnName(i) +
                                   " - Type: " + rsMetaData.getColumnTypeName(i) +
                                   " - Size: " + rsMetaData.getColumnDisplaySize(i));
            }

            // Process the result set and print data
            System.out.println("\nQuery Results:");
            while (resultSet.next()) {
                for (int i = 1; i <= columnCount; i++) {
                    System.out.print(rsMetaData.getColumnName(i) + ": " + resultSet.getStrin
                }
                System.out.println();
            }

            // Close the resources
            resultSet.close();
            statement.close();
            connection.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```