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.

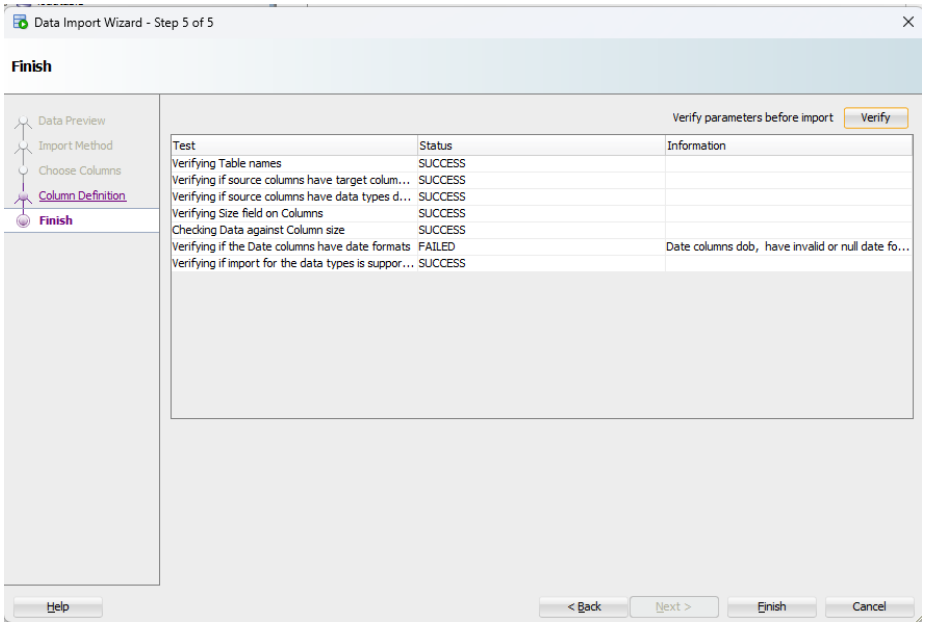

Figure 1: tableload_success

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
```

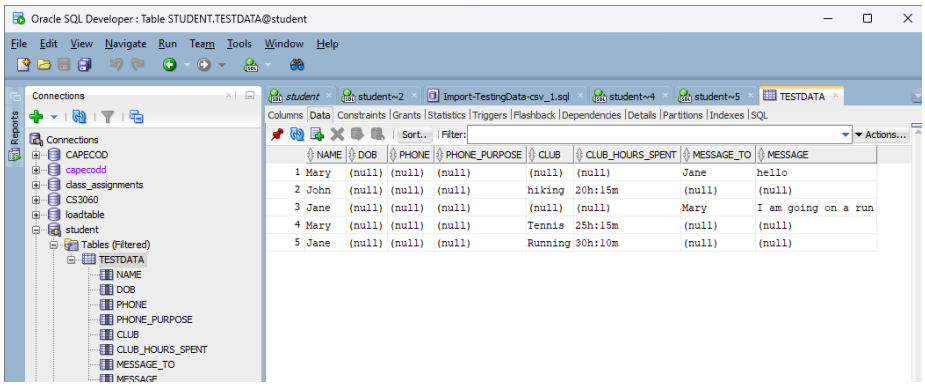

Figure 2: testdataTable

```
);

-- 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
```

```sql
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;
```

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.

```sql
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;

-- 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;
```

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));
```

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.

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.

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.

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.

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.

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.