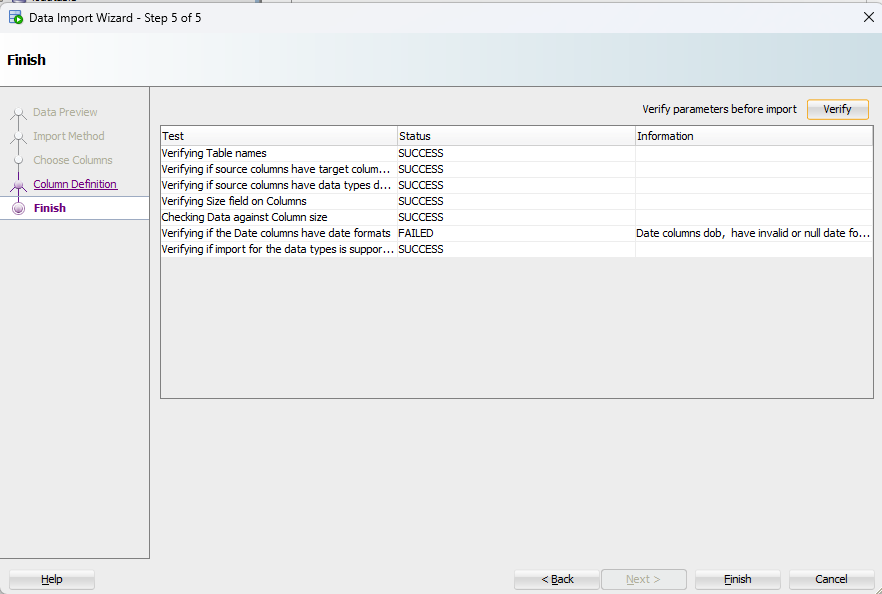
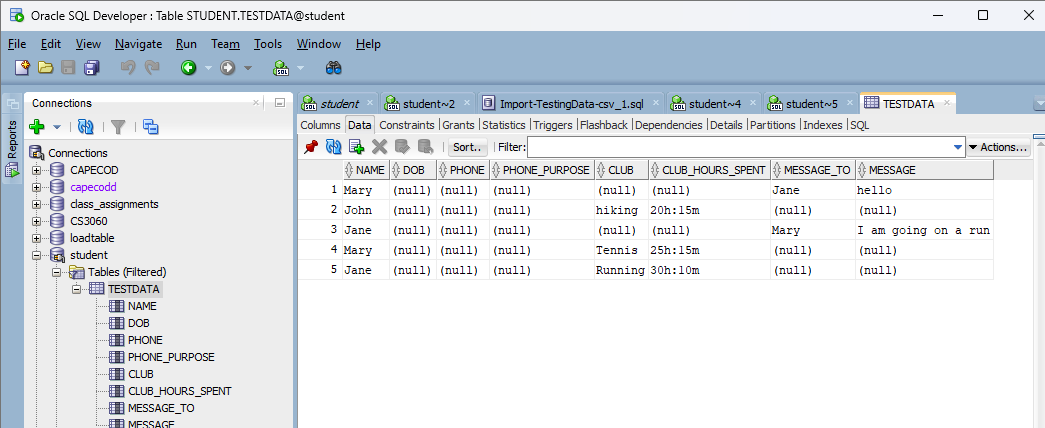
# Q1:

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.



tableload\_success



testdataTable

# 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  
);  
  
-- 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;  
  
-- 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.

--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');  
  
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.

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 `drop`  
  
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\_from and message\_to in the student\_message\_table.  
  
create or replace view student\_message\_view as  
select  
 (select name from student where student.student\_id = student\_message\_table.message\_from) as MessageFROM,  
 (select name from student where student.student\_id = student\_message\_table.message\_to) as MessageTO,  
 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) as sender,  
 (select dob from student where student.id = student\_message\_table.message\_from) as senderDoB,  
 (select phone\_number from student where student.student\_id = student\_message\_table.message\_to) as fromNumber,  
  
  
 (select name from student where student.student\_id = student\_message\_table.message\_to) as recipient,  
 (select dob from student where student.id = student\_message\_table.message\_to) as recieverDoB,  
 (select phone\_number from student where student.student\_id = student\_message\_table.message\_to) as toNumber,  
  
 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") + " " + studentColumns.getString("Type Name") + studentColumns.getString("Column Size") );  
 }  
  
 studentColumns.close();  
  
  
 while (studentColumns.next()) {  
 System.out.println(" " + studentColumns.getString("Column Name") + " " + studentColumns.getString("Type Name") + studentColumns.getString("Column Size") );  
 }  
 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 time spent  
 String query = "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";  
  
 // 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.getString(i) + "\t");  
 }  
 System.out.println();  
 }  
  
 // Close the resources  
 resultSet.close();  
 statement.close();  
 connection.close();  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}