4

CSDS 341 HW 4

1

Using MS SQL Server syntax, give a create table statement to create in the University database a table named `student2` that has the same attributes as the student relation, except make `student2.id` an integer that is automatically populated when a row is inserted.

✓ **Answer**

```
CREATE TABLE student2 (
    ID              INT IDENTITY(1,1),
    name            VARCHAR(20) NOT NULL,
    dept_name       VARCHAR(20),
    tot_cred        NUMERIC(3,0) CHECK (tot_cred ⩾ 0),
    PRIMARY KEY (ID),
    FOREIGN KEY (dept_name) REFERENCES department (dept_name)
        ON DELETE SET NULL
);
```

2

Give a single “Insert into” statement to insert into `student2` all rows from the student table. This should be a single insert statement.

✓ **Answer**

```
INSERT INTO student2 (
    name, dept_name, tot_cred
) SELECT s.name, s.dept_name, s.tot_cred
FROM student as s;
```

3

Stored procedures.

a

Give the SQL code to create a stored procedure named `insertStudent2` that takes as input the appropriate attributes to add a new row into the `student2` table and returns the identity id that is created

✓ **Answer**

```
CREATE OR ALTER PROCEDURE insertStudent2
    @name VARCHAR(20),
    @dept_name VARCHAR(20),
    @tot_cred NUMERIC(3,0),
    @id INT OUTPUT
AS
BEGIN
    INSERT INTO student2 (name, dept_name, tot_cred)
        VALUES (@name, @dept_name, @tot_cred);

    SELECT @id = SCOPE_IDENTITY();
END;
```

b

Provide an “exec” statement that can be used to execute your stored procedure in SSMS.

✓ **Answer**

```
DECLARE @idd AS INT;

EXEC insertStudent2
    @name = 'Trevor',
    @dept_name = 'Comp. Sci.',
    @tot_cred = 96,
    @id = @idd OUTPUT;

SELECT @idd;
```

4

Give the necessary sql statements to grant dbuser read/write permissions to the table `student2`.

✓ **Answer**

```
CREATE LOGIN dbuser
    WITH PASSWORD = 'Password123';


CREATE USER dbuser
    FOR LOGIN dbuser;

GRANT SELECT, INSERT, UPDATE, DELETE ON student2 TO dbuser;
```

5

Give the necessary sql statements to grant the correct permissions to the dbuser to execute the stored procedure `student2`.

✓ **Answer**

```
GRANT EXECUTE ON OBJECT::insertStudent2
    TO dbuser;
```

6

Java code using JDBC on CSDS 341 Desktop within Visual Studio code

a

Give the Java with JDBC code to execute your stored procedure and displays the identity id that is returned.

✓ **Answer**

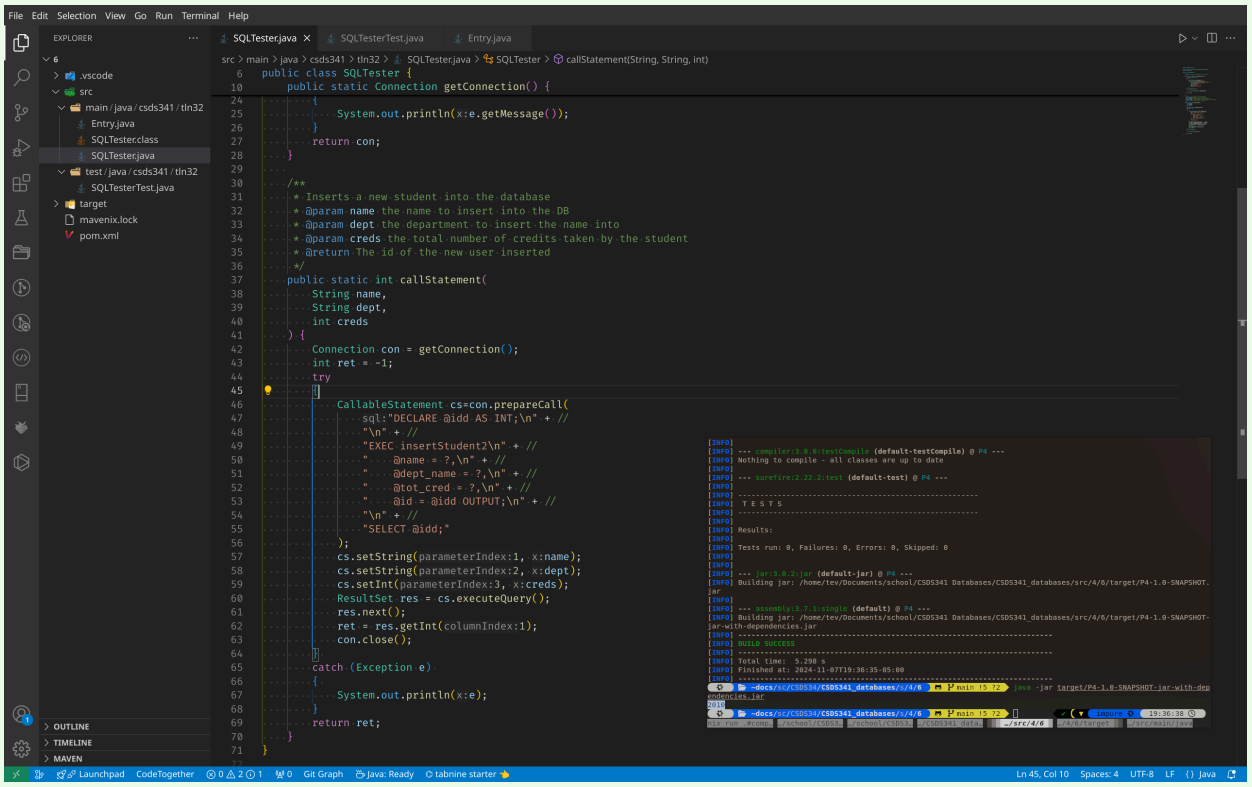
```java
package csds341.tln32;

import com.microsoft.sqlserver.jdbc.SQLServerDriver;
import java.sql.*;

public class SQLTester {
    /**
     * @return Current DB connection for this project
     */
    public static Connection getConnection() {
        Connection con = null;
        try
        {
            SQLServerDriver.register();
            con = DriverManager.getConnection(
                "jdbc:sqlserver://localhost:1433;" + //
                "databaseName=University;" + //
                "encrypt=false",
                "dbuser",
                "Password123"
            );
        }
        catch (SQLException e)
        {
            System.out.println(e.getMessage());
        }
        return con;
    }

    /**
     * Inserts a new student into the database
     * @param name the name to insert into the DB
     * @param dept the department to insert the name into
     * @param creds the total number of credits taken by the student
     * @return The id of the new user inserted
     */
    public static int callStatement(
        String name,
        String dept,
        int creds
    ) {
        Connection con = getConnection();
        int ret = -1;
        try
        {
            CallableStatement cs=con.prepareCall(
                "DECLARE @idd AS INT;\n" + //
```

```
                "\n" + //
                "EXEC insertStudent2\n" + //
                "    @name = ?,\n" + //
                "    @dept_name = ?,\n" + //
                "    @tot_cred = ?,\n" + //
                "    @id = @idd OUTPUT;\n" + //
                "\n" + //
                "SELECT @idd;"
            );
            cs.setString(1, name);
            cs.setString(2, dept);
            cs.setInt(3, creds);
            ResultSet res = cs.executeQuery();
            res.next();
            ret = res.getInt(1);
            con.close();
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
        return ret;
    }
}
```

b

Give a screenprint of your working code for part a. This should show a screenprint of your CSDS Desktop within Visual Studio Code with the correct input/output prompts/display.

✓ **Answer**



I built this locally with the Microsoft SQL Server docker container, VSCodium, and maven for java to get dependencies.

The output (2010) is at the bottom left of the terminal window on the bottom right of the screen.

7

Explain why `PreparedStatements` are preferred to `Statement` objects when coding in Java and using JDBC to access the database.

a

Differentiate between the two with respect to parameters. When would one be used over the other.

✓ Answer

`PreparedStatements` are preferred as they are precompiled, parameterized, and protect from SQL injections. `Statements` are string based and do not parameterize on their own. It is possible to dynamically create a string that gets used with a `Statement`, but this is not recommended for dynamic calls as it opens up the call to SQL injections.

b

What is a SQL Injection Attack and how to avoid it?

✓ **Answer**

An SQL injection is when a parameter gets interpreted as part of the SQL query to the SQL server and interpreted as code. This is very harmful as it allows arbitrary code from the input to be executed on the server. We can sanitize our inputs in order to avoid this by looking at quotes, semicolons, and other SQL characters. `PreparedStatement` does this by default.

8

Weak Entities question.

a

What are the conditions that make an entity a weak entity?

✓ **Answer**

An entity is a weak entity if its existence depends on another entity and its primary key is partially or totally derived from another entity's attributes.

b

Consider the transact table from assignments 2 and 3. Is this table an example of a weak entity? Why or why not? Explain. In your explanation include a comment on the `exhibitionID` and how this affects the entity as being “weak” or “strong”. Also, notice that the table uses an identity for its primary key. So, what about the relationship with the work table? Write about this too.

✓ **Answer**

It is a strong entity because its Primary Key is not based off of other entities, and has its own existence. tID is not a foreign key and is also an IDENTITY column. The exhibitionID can also be null, therefore the existence of the transaction does not depend on the exhibition. It does depend on work, however, as that Foreign Key on transact is not null.

9

a

Give an example of a weak entity that is not used in class or in the presentations. Explain your example fully as why it is a weak entity.

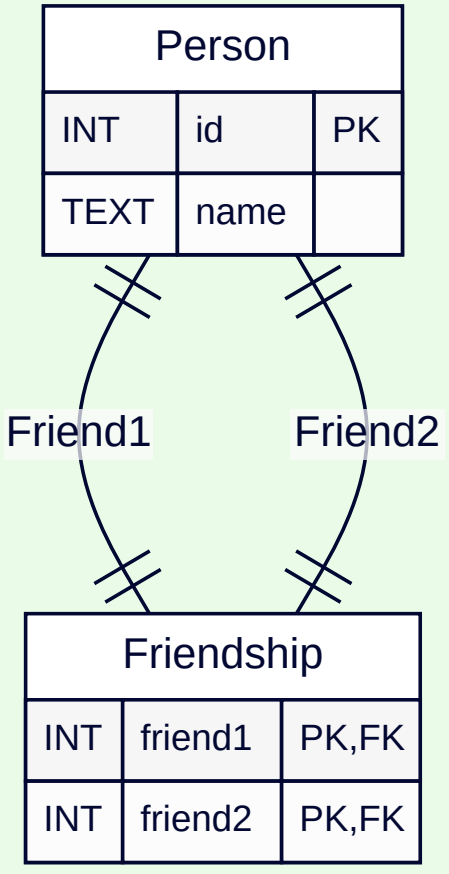
✓ **Answer**

A friendship is a weak entity as it wholly depends on the two people it connects. A friendship table may be defined as two different people's id's only, with its primary key being both ID columns.

b

Then draw both the Crow's Foot ER diagram and Chen's Foot ER diagram for your example. For the Crow's Foot ER diagram, make certain to use the following style to represent the foreign and primary keys. Again, use your own example and not one from the class, presentations or slides.

✓ **Answer**



10

Consider the Artistic Creation database from Assignment 3. Review the material from this assignment if you need to. Give a “Use Cases” for this database. This is not a single select/insert/update/delete statement but how the Artistic Creation database might be used and should include a particular scenario and at least a two of the select/insert/update/delete statements. You will be doing something similar for your projects but for your own database. You will be doing something similar for your projects but for your own database. So, for the use case:

a

Give a description/summary of the use case

✓ **Answer**

Selling a piece of art

b

Give SQL Code that your use cases will need

✓ **Answer**

```
DECLARE @wID INT;
SELECT @wID = 1500;

INSERT INTO transact (eID, patron, wID, tranDate, paid, status)
VALUES
    (6, 'Trevor', @wID, '2022-01-01', 100, 'paid');

UPDATE works
    SET status = 'sold'
    WHERE wID=@wID;
```

11

Interesting tidbits question.

a

There is a quick way to create a copy of a relation and its data using SQL. Here is an example of SQL code to do this. Using Microsoft SQL Server, run this against your

“university” database. You can use your “CSDS 341 Desktop” as SSMS is installed for you and the “university” database is loaded as well.

```
select * into newStudent from student;
```

What are the primary and foreign keys for this new table?
Aside: This simple code for creating a duplicate copy of a table is helpful in industry for testing queries quickly against a temporary table you create then later delete, you can read this article for more information if desired.

https://www.mssqltips.com/sqlservertip/6977/sql-select-into-create-table/

✓ **Answer**

The new table has no primary keys and no foreign keys, yet all the same data is present in the database as it only selects the actual data from the table, not the constraints.

b

As noted in class, Cartesian products are frowned upon especially when written as such `select * from table1, table2`. However, there are at times a need to use a Cartesian product; it is a quick way to generate MANY rows and populate a relation to do some data analysis/testing against. Modern syntax uses the key words `CROSS JOIN` as is used in a similar manner as the “inner join, left outer join, full outer join and right outer join.

Give the MS SQL Server syntax that uses the “CROSS JOIN” to create a new table that contains the Cartestian product of the artist and work relations from Assignment 2/3.

✓ **Answer**

```
SELECT * INTO artistWork FROM artist
        CROSS JOIN work;
```

12

Revisiting queries. For this question, you must give a single Microsoft SQL DML statement, and you may NOT use a “with” clause. You may use nested queries if needed. You should run and test your query using your CSDS 341 Desktop in SSMS and the “University” database. Again, your answer should be typed.

a

Write a single SQL query to return a student's `name, dept_name, tot_cred` and the average `tot_cred` of all students. Make the "average tot_ cred" attribute name be `avgCred`. You may NOT use a "with" clause.

✓ **Answer**

```
SELECT name, dept_name, tot_cred, (SELECT AVG(tot_cred) FROM
student2) AS avgCred
    FROM student2;
```

b

Write a single SQL query to return a student's `name, dept_name, tot_cred` and the average `tot_cred`, again naming this attribute `avgCred`, but this time only return rows for students with a `tot_cred` greater than or equal to the average `tot_cred`. You may NOT use a "with" clause.

✓ **Answer**

```
DECLARE @avg NUMERIC;
SELECT @avg = AVG(tot_cred) FROM student2;

SELECT name, dept_name, tot_cred, @avg AS avgCred
    FROM student2
    WHERE tot_cred ⩾ @avg;
```

c

Write a single SQL query to return the `dept_name` and the average `tot_cred` for students. Rename the average `tot_cred` as `avgCred`. You may NOT use a "with" clause.

✓ **Answer**

```sql
SELECT dept_name, AVG(tot_cred) AS avgCred
    FROM student2
    GROUP BY dept_name;
```

d

Write a single SQL query to return from the student relation, the attributes `avgCred` per department BUT this time, do not include in the average `tot_cred` computation student's with the ids `00128` or `44553`, and only list for departments whose average of `total_cred` is greater than 50, again excluding the students with IDs `00128` and `44553` in the calculation. You may NOT use a "with" clause.

✓ **Answer**

```sql
SELECT dept_name, AVG(tot_cred) AS avgCred
    FROM student2
    WHERE id NOT IN (00128, 44553)
    GROUP BY dept_name
    HAVING avgCred > 50;
```