CST2355 - Database Systems

# Assignment 1: Setting up Databases (15%)

*This assignment relates to the following Course Learning Requirements:*

CLR 1 - Plan, Prepare, Install, Configure, and Use a market-leading Database Management System, Data Modeling Engineering Tools, and Open Source Software.

CLR 2 - Develop Advanced Database Design and Normalization

CLR 3 - Develop advanced subjects and techniques of using the SQL database language.

|  |
| --- |
| **adhere to the Academic Integrity policy - no sharing, no copying, no group work**  **provide ONLY your own work.** |

## Background

You are a new employee at a startup company Van Whinkle. The company just purchased several mom and pop hotels that they want to incorporate under their Van Whinkle brand name. Because the company is new and the hotels were previously owned by small business owners, they do not have any electronic databases. Most have their data on paper. Only a few were a little bit more organized and have spreadsheets and QuickBooks.

## Pre-Assignment Instructions

1 -- Read Modules 1, 2 and 3 on Brightspace

2 -- Review the assignment support videos in the Additional Resources folder on Brightspace

3 -- Confirm Oracle Database, Data Modeler, SQL Developer, SQLPLUS are installed

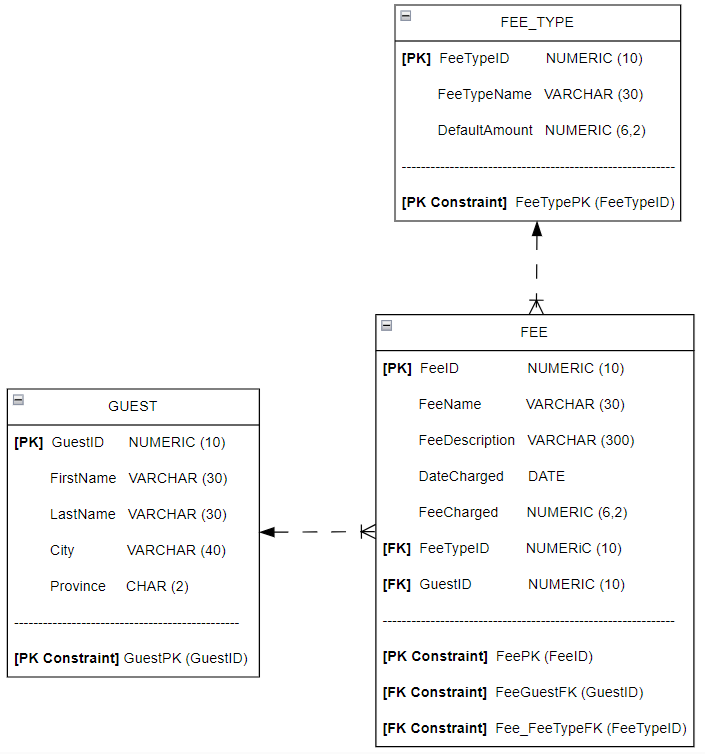
## 4 -- Review the scripts provided with this assignment

|  |
| --- |
| **Embed all screenshots in the document where requested.**  For full marks screenshots must have formatted output and complete showing ONLY the information requested. Include multiple screenshots if needed.Clear your screen using the **CLEAR SCREEN** command and use the **CST2355\_A1\_Format.SQL** script to beautify your query results. |

**Instructions**

|  |
| --- |
| The company wants to record which guests were charged which fees for which service. Based on that you will create a database design diagram (logical design) using Data Modeler. You will use the diagram to generate SQL to create your physical design for the Oracle database. Data Modeler has the option to export SQL to create the database on Oracle, SQLServer, or DB2. You use the tool to create SQL then run that SQL on the target database.  The consequence of this design is that when the name of the fee type changes, that name change is done for all fees charged. This is useful for reporting where the business uses the most recent fee type name for querying. However, that name change can cause problems with running invoices where customers may be confused by the name change. That is why there is a FeeName column in FEE.  Note that though you are creating a logical model, Data Modeler calls it a **Relational Model**. |

1 -- Create the following Relational Diagram using Data Modeler.



You must match the upper case and lower case standard and the dashed lines used in the diagram. Add a comment that includes you name, student number, and today’s date. Save your model.

Provide a screenshot of your diagram.

<replace this text with your screenshots>

2 -- Forward engineer the diagram to generate your SQL script. (You must include DROP table statements.) Remove all comments from your script. Insert the clause GENERATED AS IDENTITY for each Primary ID column before each of its NOT NULL

*FeeID NUMBER(10) NOT NULL*

*becomes*

*FeeID NUMBER(10) GENERATED AS IDENTITY NOT NULL*

Do not insert the clause for the Foreign IDs in FEE. That will result in errors.

Save your script as a file with your name and the step number. Login as ASSIGN\_1 in SQLPLUS. Clear the screen. Run your file using the @ command.

Provide a screenshot from SQLPLUS of your @ command and its error-free ran results. (Drop table messages are acceptable).

<replace this text with your screenshots>

|  |
| --- |
| Data is collected and structured to be imported into the new database structure. The structure can take the form of spreadsheets, tables, or INSERT statements. An INSERT script is provided to you by the data team. |

3 -- In SQLPLUS run CST2355\_A1\_Insert1.SQL using the @ command. This will insert data into the GUEST, FEE, and FEE\_TYPE tables. If INSERT errors occur repeat steps 1 and 2 with your corrections.

Provide a screenshot from SQLPLUS of your @ command and its error-free ran results.

<replace this text with your screenshots>

|  |
| --- |
| Because of time and coding complexity there can be omissions of data inserted into the database. This will require you to go into tables individually to add, delete, or modify data. You see none of the fees charged in the FEE table were associated to guests and fee types. You will need to do a manual update to associate these fees. |

4 -- In SQL Developer go to the FEE table’s data entry screen (data tab). Go to the FeeDescription column. You will see guest names and fee types as part of the fee description.

Using the FeeDescription column, lookup the guest’s GuestID . Put that value as the GuestID in that FEE table’s row.

Lookup the fee type’s FeeTypeID. Put that value as the FeeTypeID for that FEE table’s row.

Provide screenshots from SQL Developer of all the data stored in FEE table’s data entry screen.

<replace this text with your screenshots>

|  |
| --- |
| Databases are designed to adapt and expand by adding new tables, columns, and data constraints. Before any changes are made you make a back up the data as a series of INSERT statements. |

5 -- Use SQL Developer and its database export feature to generate INSERT statements from your existing data. Do not generate the CREATE TABLE statements.

Modify the order of the INSERT statements where parent table data is inserted before its child table data. Not doing this correctly will result in INSERT errors.

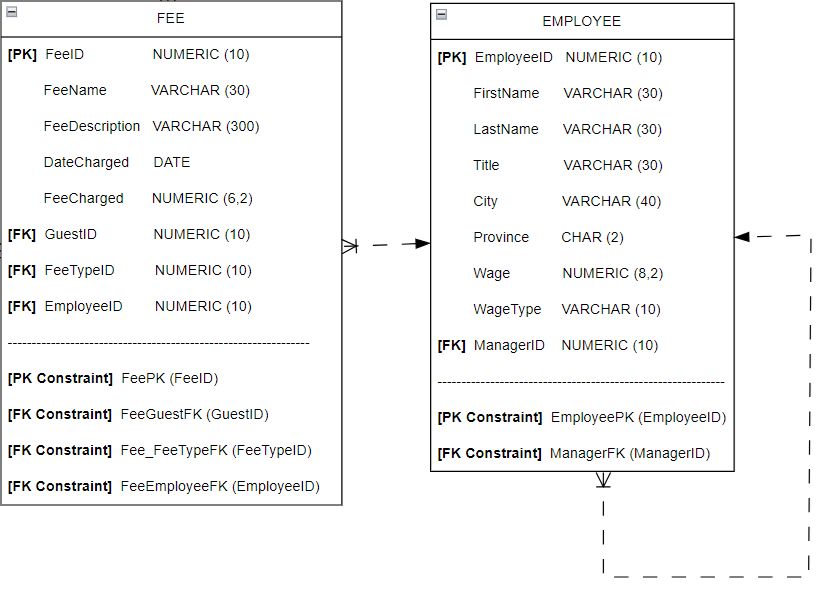
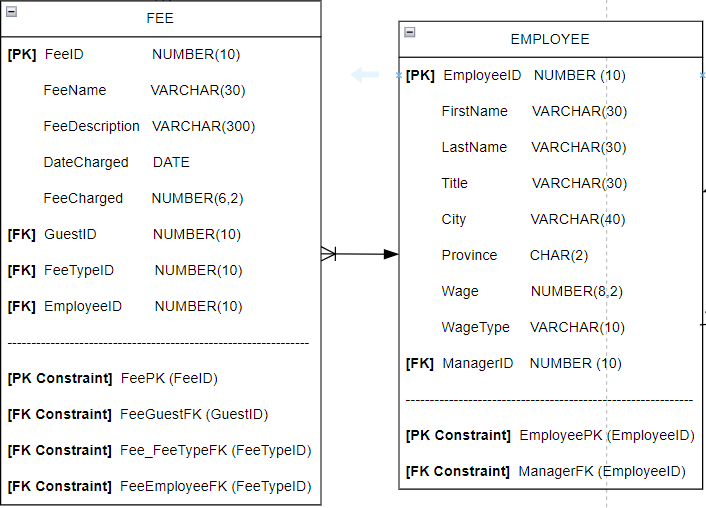
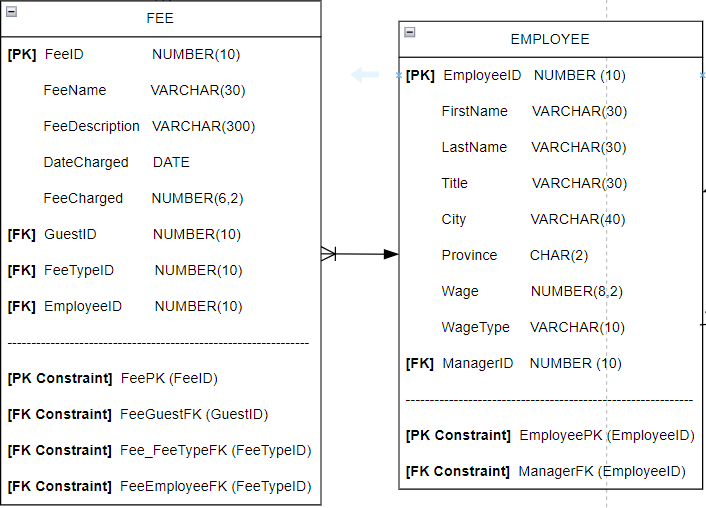
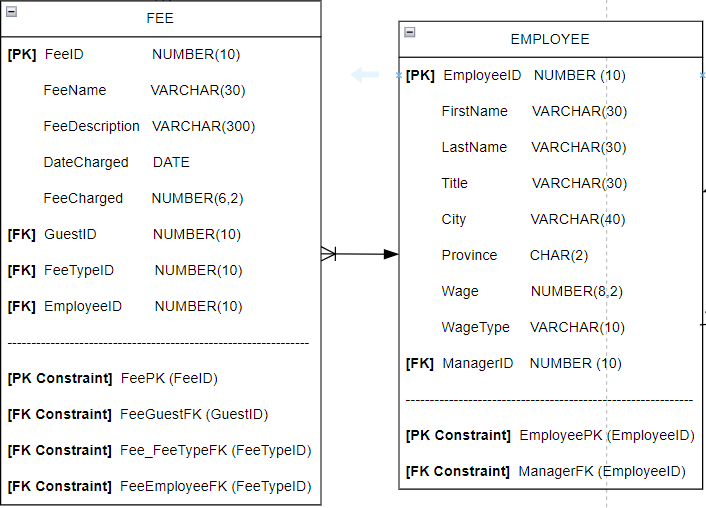
At the end of the generated script add a COMMIT statement. Save your INSERT script as a file with your name and step number.

Provide a screenshot from SQL Developer of the first page of your INSERT script.

<replace this text with your screenshots>

|  |
| --- |
| You are requested to add an EMPLOYEE table. Employees are the intermediaries that charge fees for the hotel’s services. |

6 -- Go back to your Relational Diagram in Data Modeler from step 1. Add the EMPLOYEE table and add a relationship to FEE. Your diagram must match the columns, naming standard and dash lines shown below.



Correct all new columns that are automatically added.

Save your diagram. Provide a screenshot of your complete diagram.

<replace this text with your screenshots>

|  |
| --- |
| Rather than focusing on an incremental change to the database, you will instead rebuild the database with the new design then re-insert your backed up data. This ensures your model matches what you have in your database. |

7 -- Forward engineer the diagram from step 6 to generate your SQL script. (You must include DROP table statements.) Save your script as a file with your name and the step number.

Go to SQLPLUS. Clear the screen. Run your script using the @ command to delete and replace your existing database.

Provide a screenshot of your @ command and its error-free ran results. (Drop table messages are acceptable.)

<replace this text with your screenshots>

Clear your screen. Run the your INSERT script from step 5 into SQLPLUS using copy/paste. This will put your data back into the database.

Provide screenshots of all your error-free INSERT messages.

<replace this text with your screenshots>

|  |
| --- |
| The employee-manager relationships are omitted from the EMPLOYEE table. You will now add them by hand. |

8 -- Run CST2355\_A1\_Insert2.SQL In SQL Developer. Go to the EMPLOYEE table’s data entry screen. Update each ManagerID in EMPLOYEE to establish the following employee-manager relationships.

Bob Smith (Service Manger) reports to Deena Donor

Bob Smith (Desk Clerk) reports to Bob Smith (Service Manager)

Tanya Duncan reports to Bob Smith (Service Manager)

Deena Donor doesn’t report to anyone

Provide a screenshot from SQL Developer of all data in EMPLOYEE data entry screen.

<replace this text with your screenshots>

|  |
| --- |
| It is often a good idea to check your work to confirm you added the employee-manager relationships correctly. The best way to do this is by running a query. |

9 -- Use SQL Developer to write a query that returns the data from the EMPLOYEE table in the following form. Redo step 8 if your results do not match below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SUBORDINATEFIRSTNAME** | **SUBORDINATELASTNAME** | **SUBORDINATETITLE** | **MANAGER**  **FIRSTNAME** | **MANAGER**  **LASTNAME** | **MANAGERTITLE** |
| Bob | Smith | Service Manager | Deena | Donor | Hotel Manager |
| Bob | Smith | Desk Clerk | Bob | Smith | Service Manager |
| Tanya | Duncan | Maintenance | Bob | Smith | Service Manager |
| Deena | Donor | Hotel Manager |  |  |  |

Use LEFT JOIN and table aliases to have EMPLOYEE joined to itself using its EmployeeID and ManagerID columns. Use the table alias MANAGER and SUBORDINATE. Add a comment before the query that includes your name, student number, and today’s date.

In SQLPLUS run CST2355\_A1\_FORMAT.SQL using the @ command. Clear your screen. Copy and paste into SQLPLUS to run your script.

Provide a screenshot from SQLPLUS of your comment, SELECT statement, and its **formatted**output results.

<replace this text with your screenshots>

|  |
| --- |
| Business wants to prioritize its guests so they can offer them deals and discounts. They want to know their three top revenue-generating guests. |

10 -- Use SQL Developer to write a query that returns the following results with total fees in descending order. Use only the FEE table.

|  |  |
| --- | --- |
| **GUESTID** | **TOTALFEE** |
| 1 | 870 |
| 6 | 160 |
| 2 | 30 |
| 5 | 0 |

TotalFee is the total fees charged (SUM) for each GuestID. Add a comment before the query that includes your name, student number, and today’s date.

In SQLPLUS clear your screen. Copy and paste into SQLPLUS to run your script.

Provide a screenshot from SQLPLUS of your comment, SELECT statement, and its **formatted** output results.

<replace this text with your screenshots>

|  |
| --- |
| The business doesn’t want the query results to be sorted by rank but by last name. Sorting by last name results in the rank changing and becoming incorrect. Because of how ROWNUM works you will need to make your query a subquery so your rank column values do not change. |

11 -- Go to SQL Developer. Create a new top-level query using SELECT \* FROM. Put the query in step 10 as as subquery after the FROM. The complete query should return the following results.

|  |  |
| --- | --- |
| **GUESTID** | **TOTALFEE** |
| 1 | 870 |
| 6 | 160 |
| 2 | 30 |
| 5 | 0 |

Do not give your subquery an alias. Use indenting for readability. Add a comment before the query that includes your name, student number, and today’s date.

In SQLPLUS clear your screen. Copy and paste into SQLPLUS to run your script.

Provide a screenshot from SQLPLUS of your comment, SELECT statement, and its **formatted** output results.

<replace this text with your screenshots>

12 -- Go to SQL Developer. Modify your query in step 11 to return the following results.

|  |  |  |
| --- | --- | --- |
| **GUESTID** | **TOTALFEE** | **RANKLETTER** |
| 1 | 870 | A |
| 6 | 160 | B |
| 2 | 30 | C |

|  |
| --- |
| Refer to [Oracle Tutorial - Oracle Subquery](https://www.oracletutorial.com/oracle-basics/oracle-subquery/), [Oracle Help Center - SQL Language Reference - ROWNUM Pseudocolumn](https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/ROWNUM-Pseudocolumn.html" \l "GUID-2E40EC12-3FCF-4A4F-B5F2-6BC669021726), and [Oracle PL/SQL Programming, Third Edition - CASE Statements](https://www.oreilly.com/library/view/oracle-plsql-programming/0596003811/ch04s02.html" \l ":~:text=Simple%20CASE%20statements%20take%20the,PL%2FSQL%20first%20evaluates%20expression%20.) when needed. |

Replace the \* with the correct column names. Use ROWNUM within the CASE statement in the top-level SELECT to return the rank letters. If ROWNUM = 1 return A. If 2 return B. If 3 return C.

*SELECT …,*

*CASE ...*

*WHEN … THEN …*

*…*

*WHEN … THEN …*

*END AS …*

*FROM ( … )*

Add a WHERE clause to the top-level SELECT using ROWNUM to filter by the first 3 rows.

Add a comment before the query that includes your name, student number, and today’s date.

In SQLPLUS clear your screen. Copy and paste into SQLPLUS to run your script.

Provide a screenshot from SQLPLUS of your comment, SELECT statement, and its **formatted** output results

<replace this text with your screenshots>

|  |
| --- |
| The biggest challenge to scaling database performance is the performance of queries. When processing aggregates such as totals on a large number of records you contain the processing in subqueries. You are requested to create a query that process the total charged for rate and damage fees for each guest. You will build on this query to create a much more complicated one for a report. |

13 -- Go to SQL Developer. Create a top-level query using SELECT \* FROM GUEST.

Add your query in step 12 as a subquery with the alias TOTAL\_FEE. INNER JOIN your subquery to GUEST using that subquery’s alias. Sort by last name.

Replace the \* with the correct column names. The complete query should return the following results sorted by LastName.

|  |  |  |  |
| --- | --- | --- | --- |
| **FIRSTNAME** | **LASTNAME** | **TOTALFEE** | **RANKLETTER** |
| Billy | Elliot | 30 | C |
| Justin | Hackman | 160 | B |
| Kristoff | Kurn | 870 | A |

Add a comment before the query that includes your name, student number, and today’s date.

In SQLPLUS run CST2355\_A1\_FORMAT.SQL using the @ command. Clear your screen. Copy and paste into SQLPLUS to run your script.

Provide a screenshot from SQLPLUS of your comment, SELECT statement, and its **formatted** output results.

<replace this text with your screenshots>

14 -- Go to SQL Developer. Add a WHERE clause to your subquery in step 13 to filter the fee types used by the SUM. Use a SELECT… FROM FEE\_TYPE WHERE… within the IN operator to do this. Include only fee types named rate and smoke damage. (Oracle is case sensitive. Your values must match what is in your FEE\_TYPE table.) Refer to [W3Schools - SQL IN Operator](https://www.w3schools.com/sql/sql_in.asp) for examples. The query should return the following results.

|  |  |  |  |
| --- | --- | --- | --- |
| **FIRSTNAME** | **LASTNAME** | **TOTALFEE** | **RANKLETTER** |
| Billy | Elliot | 30 | C |
| Justin | Hackman | 160 | B |
| Kristoff | Kurn | 840 | A |

If you haven’t already, add a comment before the query that includes your name, student number, and today’s date.

In SQLPLUS run CST2355\_A1\_FORMAT.SQL using the @ command. Clear your screen. Copy and paste into SQLPLUS to run your script.

Provide a screenshot from SQLPLUS of your comment, SELECT statement, and its **formatted** output results.

<replace this text with your screenshots>

|  |
| --- |
| You will now embed your query into a new query that will be used for a complex business report. The report shows a breakdown of the total fee by the fees charged. |

15 -- Go to SQL Developer and write a query that returns the following results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FIRSTNAME** | **LASTNAME** | **FEETYPENAME** | **DATECHARGED** | **FEECHARGED** | **TOTALFEE** |
| Justin | Hackman | Deposit | 23-02-02 | 50 | 160 |
| Justin | Hackman | Discount | 23-02-02 | -30 | 160 |
| Justin | Hackman | Rate | 23-02-02 | 40 | 160 |
| Justin | Hackman | Rate | 23-02-08 | 40 | 160 |
| Justin | Hackman | Rate | 23-02-17 | 40 | 160 |
| Justin | Hackman | Rate | 23-02-27 | 40 | 160 |
| Justin | Hackman | Deposit | 23-02-27 | -20 | 160 |
| Kristoff | Kurn | Deposit | 23-02-03 | 40 | 870 |
| Kristoff | Kurn | Rate | 23-02-03 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-04 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-05 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-06 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-07 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-08 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-09 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-10 | 50 | 870 |
| Kristoff | Kurn | Rate | 23-02-11 | 60 | 870 |
| Kristoff | Kurn | Rate | 23-02-12 | 60 | 870 |
| Kristoff | Kurn | Rate | 23-02-13 | 60 | 870 |
| Kristoff | Kurn | Rate | 23-02-14 | 60 | 870 |
| Kristoff | Kurn | Rate | 23-02-15 | 60 | 870 |
| Kristoff | Kurn | Rate | 23-02-16 | 60 | 870 |
| Kristoff | Kurn | Rate | 23-02-17 | 60 | 870 |
| Kristoff | Kurn | Smoke Damage | 23-02-17 | 20 | 870 |
| Kristoff | Kurn | Carpet Damage | 23-02-17 | 30 | 870 |
| Kristoff | Kurn | Deposit | 23-02-17 | -40 | 870 |
| Billy | Elliot | Deposit | 23-01-20 | 0 | 30 |
| Billy | Elliot | Rate | 23-01-20 | 30 | 30 |
| Deena | Donor | Deposit | 23-01-01 | 0 | 0 |
| Deena | Donor | Rate | 23-01-01 | 0 | 0 |
| Deena | Donor | Deposit | 23-02-01 | 0 | 0 |
| Deena | Donor | Rate | 23-02-01 | 0 | 0 |
| Deena | Donor | Deposit | 23-03-01 | 0 | 0 |
| Deena | Donor | Rate | 23-03-01 | 0 | 0 |
| Deena | Donor | Deposit | 23-04-01 | 0 | 0 |
| Deena | Donor | Rate | 23-04-01 | 0 | 0 |

Use your query in step 10 as a subquery with alias TOTAL. INNER JOIN together the GUEST, FEE, and TOTAL subquery.

Add a comment before the query that includes your name, student number, and today’s date.

In SQLPLUS run CST2355\_A1\_FORMAT.SQL using the @ command. Clear your screen. Copy and paste into SQLPLUS to run your script.

Provide a screenshot from SQLPLUS of your comment, SELECT statement, and its **formatted** output results.

<replace this text with your screenshots>

|  |
| --- |
| Confirm all screenshots are correct and not clipped. Save your assignment as a PDF and submit it in the assignment submission folder. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Assignment 1 Rubric** | | | |
| **Criteria** | **Accomplished** | **Developing** | **Incomplete** |
|  | **1 points** | **0.5 point** | **0 points** |
| 1 -- 1st Database Design | Completed all instructions and submitted correct results | Completed all instruction but submitted incorrect results | Not per instructions, incomplete, or omitted |
| 2 -- 1st Forward Engineering | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Not per instructions, incomplete, or omitted |
| 3 -- Database Design Confirmation | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 4 -- FEE Data | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Not per instructions, Incomplete or omitted |
| 5 -- INSERT Backup Script | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Not per instructions, incomplete, or omitted |
| 6 -- 2nd Database Design | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Not per instructions, incomplete, or omitted |
| 7 -- 2nd Forward Engineering | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 8 -- EMPLOYEE Data | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Not per instructions, Incomplete or omitted |
| 9 -- Manager / Subordinate | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 10 -- Total Fees Descending | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 11 -- Subquery no Alias | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 12 -- Top Three Fee Totals | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 13 -- Top Three Guests | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 14 -- Top Three Filtered | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |
| 15 -- Guests with Fees | Completed all instructions and submitted correct results | Completed all instructions but submitted incorrect results | Doesn't compile, not per instructions, Incomplete or omitted |