EAST TENNESSEE STATE UNIVERSITY

CSCI 2020 – Database Fundamentals
Lab 2 – Introduction to SQL

This lab is worth 50 total points.

Header Comments

Open a new SQL Worksheet for your connection to PYTHIA. Enter the following comments into your SQL Worksheet in SQL Developer. You will include this block of comments at the beginning of ALL SQL FILES you submit. **This, as with all submissions, will receive a 10% deduction if this header block is not included.**

```
/*
-----------------------------------------------------------------------------------------------------------
Programmer's Name: Im A. Student
Course: CSCI 2020 Section Number: 940
Creation Date: 01/28/2021 Date of Last Modification: 01/28/2021
E-mail Address: studentia@etsu.edu
-----------------------------------------------------------------------------------------------------------
Purpose -
    Lab 2 - Introduction to SQL
-----------------------------------------------------------------------------------------------------------
Identifier dictionary -
    Not Applicable
-----------------------------------------------------------------------------------------------------------
Notes and Assumptions -
-----------------------------------------------------------------------------------------------------------
*/
```

In Oracle, single-line comments begin with dash-dash (`--`). This behaves exactly like Java's slash-slash comments (//). Multi-line comments in Oracle are between `/*` and `*/`, as they are in Java.

Save your file (Click menu option File > Save) as LastnameFirstinitial_Lab2.sql (e.g., RezwanaT_Lab2.sql).

A. Database Structure

ALLPOWDER is a company that specializes in the sale of ski equipment. Their database consists of the following tables that are already populated with data.

1. BindingStyle
2. Customer
3. CustomerSkill
4. Department
5. Employee
6. Inventory
7. ItemModel
8. Manufacturer

9. PaymentMethod
10. ProductCategory
11. Rental
12. RentItem
13. Sale
14. SaleItem
15. SkiboardStyle
16. SkillLevel

B. LESSON: Querying Basics

A SQL statement can span multiple lines. SQL statements are **not** case-sensitive. However, capitalizing keywords helps us differentiate between keywords and attribute/table names or values.

The basic format of a query is as follows:

```
SELECT [DISTINCT] * FieldName(s)
  FROM TableName
[Join Clauses]
[WHERE Condition(s)]
[GROUP BY List]
[HAVING AggregateCondition]
[ORDER BY List];
```

Note: the clauses within square brackets [] are optional.

For example, to view every customer's last name in the Customer table, you would enter the command as shown below:

```
SELECT lastname
FROM ALLPOWDER.Customer;
```

Note: the suffix ALLPOWDER is required here because the tables we are querying belong to a different schema in SQL Developer. In SQL Developer, a schema is a collection of database objects (tables, indexes etc.). A schema is owned by a database user and has the same name as that user. If you are querying tables that belong to your schema, this prefix will not be required.

SELECT clause

To view more than one column (field) of data in a table, you can specify the fieldnames in the Select clause, separated by commas:

```
SELECT lastname, firstname, phone
FROM ALLPOWDER.Customer;
```

To view all of the columns in a table, you can use the asterisk (*) as a wildcard character that represents all columns:

```
SELECT *
FROM ALLPOWDER.Customer;
```

Aliases in the SELECT clause

Did you notice that lastname looks really "techie"? If you were to create a report that displayed lastname as the title of a column, the report would not look very professional. Instead, you can make your query display this field as a different name (i.e. an "alias") using the keyword "AS":

```
SELECT lastname AS "Last Name"
FROM ALLPOWDER.Customer;
```

Note: If you have a space in your alias, you must enclose the alias in double quotes.

DISTINCT keyword in the Select Clause

Sometimes a table includes multiple occurrences of the same value in a field (not a primary key field, of course), and you only want to see each occurrence listed one time in your output. Assume we want to see a unique listing of all of the states in the Customer table. We can use the DISTINCT keyword to ensure that every state is only listed once:

```
SELECT DISTINCT state
FROM ALLPOWDER.Customer;
```

Aggregate operators in the Select Clause

In many queries, you are not interested in the actual values of each individual row in a table, but rather a summary of them. For example, if someone asked you how many customers had ever ordered from you, you could use the COUNT operator in your select clause. For example:

```
SELECT COUNT (CustomerID)
FROM ALLPOWDER.Customer;
```

This query would return the count (or the total) number of rows in the Customer table. Note that whenever you specify any other field in the Select clause other than the item that you are aggregating, you must also include a GROUP BY clause at the bottom of your query. For example, assume you want to know how many customers are in each country in your table, you can count the number of customers and group them by Country:

```
SELECT state, COUNT(CustomerID) as "Customers by State"
FROM ALLPOWDER.Customer
GROUP BY state;
```

Count is not the only aggregate operator available to you. SUM, AVG, MIN, and MAX are a few examples of others that you can use. Note, however, that statistical functions (e.g. SUM, AVG, etc.) can only be used on fields that contain numbers and are defined with a numeric data type. For instance, it would not make sense to SUM the last name field. However, it would make sense to SUM the QuantitySold of a particular product in the SaleItem table:

```
SELECT SUM(QuantitySold)
FROM ALLPOWDER.SaleItem
WHERE sku = 800434;
```

This query produces the sum of the quantity sold field for the product whose SKU (ID) is 800434. Note that this query includes the use of a WHERE clause (discussed next).

You can use mathematical operators (e.g. +, -, /, *) to perform calculations between the results of aggregate functions. For example, the statement below calculates the difference between the average rent fee and the average repair charge in the RentItem table.

```
SELECT AVG(rentfee) - AVG(repaircharges)
FROM ALLPOWDER.RentItem;
```

WHERE Clause

What if you want to view only those customers that meet a certain criterion? This is where the "Where" clause comes in. Let's assume you want to view customers that live in Texas:

```
SELECT *
FROM ALLPOWDER.Customer
WHERE state = 'TX';
```

Note: the value TX is encased with single quotes because it is not a numerical value but a character value.

Multiple conditions in a WHERE clause

What if you want to narrow down your results even more? Let's assume you want to view customers that live in the state of Texas AND the city of Dallas?

```
SELECT *
FROM ALLPOWDER.Customer
WHERE State = 'TX' AND City = 'Dallas';
```

What if you want to expand your results to include everyone in Texas OR New York?

```
SELECT *
FROM ALLPOWDER.Customer
WHERE state = 'TX' OR state = 'NY';
```

Note that you can combine AND's and OR's as much as needed to satisfy your question.

IN keyword

Assume you want to specify an elaborate OR condition like this:

```
SELECT *
FROM ALLPOWDER.Customer
WHERE City = 'Los Angeles' OR City = 'San Antonio'
OR City = 'San Francisco' OR City = 'Lincoln'
OR City = 'Houston' OR City = 'Richmond' OR city = 'Detroit'
OR City = 'Miami' OR City = 'Bristol';
```

SQL gives you a shortcut that eliminates much of the typing. This shortcut uses the keyword "IN". The list of items following the IN keyword is the list of items for which you are filtering:

```
SELECT *
FROM ALLPOWDER.Customer
WHERE City IN('Los Angeles','San Antonio', 'San Francisco',
'Lincoln', 'Houston', 'Richmond', 'Detroit', 'Miami', 'Bristol');
```

Concatenation Operator

|| or concatenation operator is used to link columns or character strings. We can also use a literal. A literal is a character, number or date that is included in the SELECT statement. For example:

```
SELECT employeeID, lastname||firstname as "Employee Name"
  FROM ALLPOWDER.Employee;
```

You can also use literals as demonstrated by the example below:

```
SELECT employeeID, lastname|| ' ' || 'has phone no.' || ' ' ||phone
  FROM ALLPOWDER.Employee;
```

You can also use the concatenation operator to combine the results of an aggregate function on multiple columns.

```
SELECT COUNT(state||city||zip)
FROM ALLPOWDER.Manufacturer;
```

C. Assignment (50 Points)

1. Write a select statement that returns all records from the ALLPOWDER.Employee table. Insert the statement into your SQL file. Format your answer as follows. (2 points)

```
-----------------------------------------
-- A1. Select Employee Table

<Insert Select Statement Here>
```

2. Write a select statement that returns the LASTNAME, FIRSTNAME, and DEPARTMENT of all Employees. Format your answer as follows. (3 points)

```
-----------------------------------------
-- A2. Select Employee Table

<Insert Select Statement Here>
```

3. This question has three parts. (9 Points)
    a. Write a select statement that returns the LASTNAME, FIRSTNAME, and DATEOFBIRTH of all Customers whose first name is Andy. Format the output as you did previously, but with the following heading. (3 points)

```
-----------------------------------------------------
-- A3a. Select Customer Table for Andy
```

    b. Write a select statement that returns the LASTNAME, FIRSTNAME, and EMAIL of all Customers whose first name is Sarah. Format the output as you did previously, but with the following heading. (3 points)

```
-----------------------------------------------------
-- A3b. Select Customer Table for Sarah
```

c. Write a select statement that returns the LASTNAME, FIRSTNAME, and CITY of all Customers whose last name is McCoy. Format the output as you did previously, but with the following heading. (3 points)

```
-------------------------------------------------------
-- A3c. Select Customer Table for McCoy
```

4. This question has three parts. (12 Points)
   a. Write a SELECT statement that lists the first name and last name of all Employees in the Boards department. Include the following heading. (3 points)

```
-------------------------------------------------------
-- A4a. Select Employees in Boards Dept.
```

   b. Write a SELECT statement that lists the first name and last name of all Employees in either the Ski-Alpine OR Ski-Nordic departments. Include the following heading. (3 points)

```
-------------------------------------------------------------
-- A4b. Select Employees in Ski Departments
```

   c. Write a SELECT statement that lists the first name and last name of all Employees in BOTH the Ski-Alpine AND Ski-Nordic departments. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A4c. Select Employees in both Ski Departments
```

   d. Write a SELECT statement that uses the IN keyword to list the name of all Manufacturers that produce either Boots, Clothes, Boards, Ski, Glasses or Poles. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A4d. Select Manufacturers that produce specific types of items
```

5. This question has three parts. (9 Points)
   a. Write a SELECT statement that lists ALL of the City, State and ZIP codes from the Manufacturer Table. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A5a. Manufacturer Cities
```

   b. Write a SELECT statement that lists the City, State and ZIP codes from the Manufacturer Table without duplicates. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A5b. Manufacturer Cities without Duplicates.
```

   c. How many duplicates shipping addresses (City, State and ZIP) were in the Manufacturer table? Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A5c. Number of Manufacturer Address Duplicates: ###
```

6. This question involves the use of the aggregate functions COUNT, AVG, and SUM. (15 Points)
   a. Write a SELECT statement that counts the total number of rentals stored in the system. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A6a. Number of Rentals: ###
```

b. Write a SELECT statement that counts the total number of rentals stored in the system for each payment type. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A6b. Number of Rentals by Payment Type
```

c. Write a SELECT statement that gives the total amount of the sales stored in the system (Use the SaleItem table). Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A6c. Total Amount of Sales: $###,###.##
```

d. Write a SELECT statement that gives the average sale price for all items sold. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A6d. Average Sale Price: $###,###.##
```

e. Write a SELECT statement that gives the average Sales Tax amount per Payment Type. Include the following heading. (3 points)

```
-------------------------------------------------------------------
-- A6e. Average Sales Tax Per Payment Type: $###,###.##
```

Submit your file, named LastnameFirstinitial_Lab2.sql, to the Lab 2 Dropbox area on D2L by **Wednesday, February 2 2022, 11:59 PM**.