Lab 2. Multi-table Joins and Set-based Operations

Submission:

- ~~If you decide to skip the lab, make sure you submit the **check-off questions** (highlighted with green background) in the text box of the check-off assignment item for Lab 2 before **Wednesday, noon to Brightspace**.~~
- **Lab attendance and check-off questions are waived for everyone this week.**
- **All students are expected to submit your answers to all lab questions in a text document with the name <lastname>_<firstname>_lab2.sql by the due date to Gradescope.**
- Please include both your code and the results in the **.sql** documents for full credits. For detailed requirements, please refer to the Submission Guideline on Brightspace.

Please use Eagle database for all questions in this lab.

Objectives:

- Practice performing multi-table queries and interpreting the results
- Practice performing outer joins (left join, right join, full outer join)
- Practice performing set-based operations (minus, intersect, union, union all)

Notes

- Hard code only those values explicitly stated in the problem. Never hard-code values determined dynamically.
- Do NOT include "extra" tables or columns in any query. It reduces query efficiency and increases the likelihood of error. The columns that should be displayed are underlined.
- Submissions that fail to follow the format will receive a 50% penalty.
- This lab is worth 50 points in total. You earn 25 with the no-zero policy, and the other 25 will be earned by your submitted answers.

Questions

Q#	Pts	Question Description
Inner Join with Reasoning		
1	0.5	For each part in the Accessories category (CategoryID is 'ACCESS') of the INVENTORYPART table, display its part number, part description, and the average quantity sold (OrderQuantity) of all orders placed for that part. Round the average to 1 decimal place. Sort the results by average quantity in descending order.
2a	0.5	For each month in which part 'DVD-001' was ordered, display its order month, order year, and the average quantity (OrderQuantity) sold **during that month**. Round the quantity to 1 decimal place. Sort the results first by year, then month.
2b	0.2	Based on the results of question 2a above, briefly explain how the average OrderQuantity changed over the months.
3a	0.5	For each month in which part 'DVD-001' was ordered, display its order month and year (as one column, in the format of '01-2022'), and the total quantity (OrderQuantity) sold **during that month**. Round the quantity to 1 decimal place. In addition, sort the results by chronological order. (e.g., …11-2010, 12-2010, 01-2011…)
3b	0.2	Based on the results of question 3b above, briefly explain how you would plan the procurement of part 'DVD-001' for the rest months of 2011?
4	0.5	For each month in which part 'DVD-001' was ordered, display its order month, order year, and the number of orders placed **during that month**. Sort the results first by year, then month.
5a	0.2	Explain the relationship between questions 2, 3, 4. What is the shared, underlying question that each is attempting, at least in part, to answer?
5b	0.2	Based on the answers to questions 2, 3, 4, what can we determine about the sales of part 'DVD-001'?
5c	0.2	Do the answers to question 2, 3, 4 support or conflict with each other? Does this increase or decrease our confidence in the results?
6a	0.5	For order ID '2000000007', display the order ID, shipment ID(s), package numbers, and shipped date. Also include the name of the person (ShipName) and the shipping address (ShipAddress) to which each shipment has been sent.
6b	0.2	Briefly explain the results of question 6a above.
Outer Join		
7a	0.25	Find the residential customers (whose company name is null) from Pennsylvania (state is 'PA') and all orders they have placed. Display their names in last name,

		(comma) first name format (e.g. Simpson, Lisa), customer ID, and order ID. **Using a left outer join** for this question. NOTE: Your results should include all Pennsylvania residential customers even if they have not placed an order.
7b	0.25	Find the residential customers (whose company name is null) from Pennsylvania (state is 'PA') and all orders they have placed. Display their names in last name, (comma) first name format (e.g. Simpson, Lisa), customer ID, and order ID. **Using a right outer join** for this question. NOTE: Your results should include all Pennsylvania residential customers even if they have not placed an order.
8	0.5	Display the part number and category name for all parts and all categories in the INVENTORYPART and CATEGORY tables **regardless of any missing information**.
9a	0.2	For order ID '2001000807', display the customer name in first name (space) last name format (e.g. Lisa Simpson), customer ID, and the order date. **Regardless of whether the order has been shipped**, display all shipment ID(s), package numbers assigned, the name to which each package is to be (or has been) sent (shipname), and the date on which it was sent (shippeddate).
9b	0.5	For all orders that haven't been shipped (without shippeddate), display the customer name in first name (space) last name format (e.g. Lisa Simpson), customer ID, and the order date, shipment ID(s), and the name to which each package is to be (or has been) sent (shipname).
Set Based Operation		
10a	0.5	Use an INTERSECT statement, display distinctly the customer ID of any Pennsylvania (state is 'PA') customer who has placed an order.
10b	0.5	Use a MINUS statement, display distinctly the customer ID of any Pennsylvania (state is 'PA') customer who has never placed an order.
10c	0.5	Use an INTERSECT statement, display distinctly the customer ID of any Pennsylvania (state is 'PA') customer who placed an order in 2011.
10d	0.5	Use a MINUS statement, display distinctly the customer ID of any Pennsylvania (state is 'PA') customer who did NOT place an order in 2011. (Hint: the number of rows returned by Q8a, Q8b, Q8c, Q8d should match in the following way: Q8a + Q8b = Q8c + Q8d)
11a	0.5	Display distinct part number of any cable part (CategoryID is 'CAB') which has been ordered at least once. Use **CUSTORDERLINE** table to determine if a part has been ordered or not.
11b	0.5	Display distinct part number of any cable part (CategoryID is 'CAB') which has never been ordered.

11c	0.5	Display distinct part number of any cable part (CategoryID is 'CAB') which was ordered at least once since 2010.
11d	0.4	Display distinct part number of any cable part (CategoryID is 'CAB') which was never ordered since 2010. (Hint: the number of rows returned by Q9a, Q9b, Q9c, Q9d should match in the following way: Q9a + Q9b = Q9c + Q9d)
12a	0.5	Display the first name and last name for any Florida customer (state is 'FL') in CUSTOMER table as well as the first name and last name for all Eagle employees in EMPLOYEE table. The results should **include only distinct records**. Sort the results by first name, then last name in ascending order.
12b	0.5	Display the first name and last name for any Florida customer (state is 'FL') in CUSTOMER table as well as the first name and last name for all Eagle employees in EMPLOYEE table. The results should **also include the repeating records**. Sort the results by first name, then last name in ascending order.
13a	0.5	Find all customers (including both residential and commercial customers) from Pennsylvania (state is 'PA') and all orders they have placed. Display their names (for residential customers, display customer names in the format "John Doe, residential"; for commercial customers, display customer names and the company name in the format "John Doe, Google"), customer ID, order ID, and order date. Sort the results by customer ID first, then by order ID. *Note: Your results should include all Pennsylvania customers even if they have not placed an order. Please use the UNION clause.*
13b	0.2	Please retrieve the same information as 13a ***without*** using the UNION clause.
14a	11	The goal of this problem is to generate a lot of non-repeating data from a relatively small collection. With this in mind, you are to develop a script whereby you can generate a minimum of 5 million (5,000,000) unique rows of **test data** for a table containing data on the contact person for each of our customers. Name the table: **Lab2 _CONTACT** The table should have the following attributes (columns): • First Name • Last Name • City • State You may take your initial names and locations from Eagle Database or any other source that you find, but you must document where you take it from and how you use it.

		The data used to populate the table should be appropriate to the column, NOT gibberish or random numbers. For example: city should be 'Lafayette', 'Greenville', 'Riverdale' etc., NOT 'asdf;lkjadsf' or '23adsf898'. Your answer should include all DDL and DML statements in the specific sequence necessary to generate and populate the **Lab2 _CONTACT** table. Include sufficient directions to the grader (in comment statements) to allow him/her to run your test-data generation process. **Notes:** • if your **Lab2 _CONTACT** table contains <u>less than 100,000 rows of unique data</u> you will receive NO POINTS (e.g., no partial credit) for this question! • If your **Lab2_CONTACT** table <u>was generated from hard coding</u>, you will receive a 20% deduction of the maximum available points. Use of a 3rd party tool to generate your data is NOT acceptable. You are fully equipped at this point to generate this data using SQL.
14b	3	Write one or more queries to prove that the table you created in 1a has at least 5 million rows of <u>unique</u> data. Show the number of unique First Names, unique Last Names, unique States and unique Cities. Include both the queries and your results in the submission.