Assignment 3

Writing SQL Queries

Assigned: Sep 25, 2024
Due: Oct 9, 2024. 11:59:59PM
Total Points: 100

Objectives

The purpose of this assignment is practice writing SQL queries against a particular database. On this assignment, you should only write SQL that conforms to the SQL Standard, with the exception of date/time processing, which should follow the SQLite dialect of SQL. You will be provided with an example SQLite database to test your queries against. SQLite is fairly standards compliant, so "correct" queries should work with it without modification.

Submit your solution to GradeScope before due date above. If you have further questions, please contact the instructor or any TA.

Background

Suppose you work at a distributer as a data analyst. Your main job is to analyze the data stored in their database. The database has nine tables, the schemas of which are outlined below. Primary key attributes are underlined and foreign keys are noted with a superscript, referring to the table they reference.

- address = {addr_id, addr_street, addr_state, addr_city, addr_postal_code, addr_postal_code_ex }
- item = {itm_id, itm_name, itm_unit_price, itm_vendor_sku}
- contact = {cntct_id, cntct_phone_number, cntct_phone_area_code, cntct_email_addr, cntct_firstname, cntct_lastname, cntct_job_title}
- customer = {cstmr_id, cstmr_name, cstmr_primary_contact$^{\text{FK-contact}}$, cstmr_secondary_contact$^{\text{FK-contact}}$, cstmr_addr_id$^{\text{FK-address}}$}
- invoice = {inv_id, inv_cstmr_id$^{\text{FK-customer}}$, inv_date, inv_payment_status_cd$^{\text{FK-co\_payment\_status}}$}
- line_item = {litm_inv_id$^{\text{FK-invoice}}$, litm_itm_id$^{\text{FK-item}}$, litm_quantity}
- warehouse = {wh_id, wh_addr_id}
- stock = {stk_wh_id$^{\text{FK-warehouse}}$, stk_itm_id$^{\text{FK-item}}$, stk_quantity}
- co_payment_status = {payment_status_cd, payment_status_desc }

Notes.

- All of the ID fields, which are named in the form `PREFIX_id`, are unique integers generated by the system, and not natural keys like SSNs, etc..
- The `address` table only stores addresses within the United States.
- The `address.addr_postal_code_ex` attribute stores the ZIP+4 information, and is NULLABLE.
- The `address.addr_state` attribute stores the two-letter state code for the address, and is NULLABLE.
- The `customer.cstmr_secondary_contact` attribute is NULLABLE.
- An invoice conists of line items, one per item purchase. The quantity of item purchased is carried by the `line_item` table and the cost of the item itself (per unit quantity) is on the `item` table. To get the total cost of the line item, these two numbers must be multiplied. The total cost of the invoice is the sum of the cost of all line items.

- The `inv_date` attribute has a value domain of DATE, and can be manipulated using SQLite's date and time functions. See the section of the manual titled 'Date And Time Functions' for more details on these. N.B., date manipulation is one area where various database vendors have wildly different implementations. It is *very* important that you use the SQLite manual for looking up information about this topic, rather than web search or large language models.
- *All* attributes on the `contact` table, other than `cntct_id`, are potentially NULL.
- Payment status codes are 3 character codes detailing payment status information. These are carried, along with a text description of the meaning of the code, on the `co_payment_status` table.

Questions

Write SQL queries to return the data specified in questions 1 to 23 (each is worth 4 pts.) that satisfy the following requirements:

- The answer to each question should be a single SQL query.
- You must order each query as described in the question, order is always ascending unless specified otherwise
- Every column in the result should be named. So if the query asks you to return something like income times 10, make sure you include an `AS` statement to name the column.
- While your queries will not be graded on efficiency, points may be deducted if unnecessary tables are included in the query (e.g., including both `invoice` and and `customer` when you only require the `cstmr_id` of customers on an invoice).
- **The query questions are broken up into three sections, each of which include certain restrictions upon which SQL keywords you are allowed to use. Additionally, specific questions may introduce their own restrictions. You must follow the specific requirements for each question.**

Questions 24 and 25 require the writing of DDL (also 4 pts. each). Question 25 will require more than one SQL statement in its answer.

For the following questions, you may only include a single table in the FROM clause, and you cannot use JOIN keywords or sub-queries. You may use multiple queries connected with set operators (UNION, INTERSECT, EXCEPT).

1. Return the address id for all addresses associated with *both* a warehouse and a customer without a secondary contact. You may assume that the address table is properly de-duplicated (i.e., the same address cannnot appear twice under different IDs).

2. Find all invoice IDs associated with invoices with a line item that has a quantity of between 15 and 25 items (inclusive).

3. Find the `cntct_id` for all contacts that are either the primary or secondary contact of a customer with an `cstmr_addr_id` between 5 and 10.

4. Find the contact id for all contacts who are the primary contact of one company, and the secondary contact of another different company. Order the IDs in descending order,

5. For each invoice with at least 50 items total, return its ID and the total quantity of products included on the invoice.

For the following queries, you may not use JOIN keywords or sub-queries, but you may have multiple tables listed in the FROM clause.

6. Generate a report containing the `inv_id`, total number of line items, and total cost, for all unpaid or deferred invoices with at least 2 line items, in descending order of total cost.

7. Answer the same query as from Question 4 without using any set operations.

8. Find the `cstmr_id` for all customers who are located in Pennsylvania, or have no phone number on file for both primary and secondary contacts.

9. Return the `cstmr_id`, the state in which the customer is located, and the total cost of all invoices for that customer for all customers who have a primary contact whose first name begins with 'Jess'

10. Return the first and last names of the primary contact for all customers who have unpaid invoices from greater than two years ago. For simplicity, you may assume that all years have exactly 365 days and that no dates in the database are prior to 1/1/1970.

11. Return the `cntct_id` for all contacts not associated with a customer.

12. Return the `cstmr_id`s, `addr_state`, and `wh_id` for all customers with unpaid invoices for items that are currently in stock at a warehouse in the same state.

13. Return the `cstmr_name` associated with all customers who have a primary contact that has the string 'apple' somewhere in the domain of their email address.

14. Return the `cstmr_id`s and the total outstanding invoice balance for all customers, where an "outstanding" invoice is one that is currently unpaid, or in deferment.

15. Return the total balance, from all customers, for invoices that are more than three years old and unpaid. You can assume that a year has exactly 365 days and that no dates in the database are prior to 1/1/1970.

16. Return the `inv_id` and average cost per item associated with the five invoices with the largest average cost per item (i.e., the total cost of the invoice divided by the quantity of items on the invoice).

17. Return the total amount all customers have been invoiced for 'Nvidia H100' GPUs.

18. Return the total unpaid invoice balance for all customers associated with 'Aliya Khan' (i.e., with Aliya as either the primary or secondary contact), excluding those customers with an unpaid balance less than $20,000.

19. Return the `itm_id` for all items that are currently out of stock in all warehouses.

For the following queries, you *are* allowed to use JOIN keywords and sub-queries, as well as anything allowed above.

20. Find the `cstmr_id` and the first name of the secondary contact for all customers that have purchased more than 3 units of 'V.F.D.' in the past two years. If the customer has no secondary contact, they should still appear but with a NULL value for first name.

21. Return the `litm_inv_id` and `litm_itm_id` for all invoice line items that have a quantity of greater than 2x larger than the average quantity of all line items.

22. Return the `inv_id` for the unpaid invoices with the maximum total value. Note, there may be more than one invoice satisfying this condition.

23. Return the `itm_id` and `itm_vendor_sku` for all items that are currently out of stock in exactly two warehouses. You may not join the item table as part of this query–use a sub-query instead.

The following questions require DDL statements, not queries, for their answer.

24. Write the DDL query for creating the `item` table, where the primary key and foreign key constraints must be explicitly stated. Assume that all IDs are integers, no item name is longer than 120 characters, and no SKU is greater than 30 characters. You must ensure that the unit price is a dollar amount, precise to the nearest cent ($0.01), always greater than zero, and no more than $100,000.

25. A change request has come in from the user of the system. They would like the database to track vendor information as well. Write the DDL for a new table, `vendor`, which tracks each vendor's name, address, primary contact, and the ISO 4217 currency code associated with their primary currency, as well as giving each vendor a unique integer ID number. Each item in the item table should also be associated with the vendor that provides it. Provide the necessary `ALTER TABLE` statements to make these changes. Be sure to include any necessary foriegn key constraints to both new and existing tables. The DBMS should automatically throw an exception if the vendor for items currently on the item table is deleted, and should update the ID on the item table the corresponding vendor ID is updated.