**Aaron McCarley**

**Assignment 3**

**CS300**

**Database Management Systems**

**Queries**

1. Retrieve the first name and last name of each author in the author relation. Order does not matter. **(1 points)**

**SQL Statement:**

**SELECT lastName, firstName**

**FROM author;**

**Output: Table

Description automatically generated**

1. Retrieve the title and book type in the book relation. Order does not matter. **(1 points)**

**SQL Statement:**

**SELECT title, TYPE**

**FROM book;**

**Output:**  **Table

Description automatically generated**

1. Retrieve the publisherCode in the book relation. List each publisherCode only once in the result. Order does not matter. **(1 points)**

**SQL Statement:**

**SELECT DISTINCT(publisherCode)**

**FROM book;**

**Output: Table

Description automatically generated**

1. Retrieve the title and price of each book in the book relation. Further add a calculated column named ‘discount’ that shows the price the book with 25% discount. Order does not matter. Show the first five rows of the result. **(2 points)**

I used .75 times the amount to calculate the 25% discount

**SQL Statement:**

**SELECT title, price, (price \* .75) AS discount**

**FROM book;**

**Output:** **Table

Description automatically generated with medium confidence**

1. Retrieve the title and price for any book whose price is higher than $20.00 in the book relation. Show the full result. **(2 points)**

**SQL Statement: SELECT \* FROM book WHERE price > 20;**

**Output:** **Table

Description automatically generated**

1. Retrieve the publisherName of all publishers that are in New York only in the publisher relation. Order does not matter. **(2 points)**

**SQL Statement:**

**SELECT publisherName, city**

**FROM publisher**

**WHERE city='New York';**

**Output: Table

Description automatically generated**

1. Retrieve the publisherName of all publishers that are not in New York in the publisher relation. (use != for inequality). Order does not matter. Show the full result. **(3 points)**

**SQL Statement:**

**SELECT publisherName, city**

**FROM publisher**

**WHERE city != 'New York';**

**Output: Table

Description automatically generated**

1. Retrieve the bookCode and onHand for each book for which a branch has between 2 and 4 copies in the inventory relation. **Use the BETWEEN keyword in this query.** Order does not matter. Show the full result. **(3 points)**

**SQL Statement:**

**Select bookCode, onHand**

**FROM inventory**

**WHERE onHand BETWEEN 2 AND 4;**

**Output:**

**Table

Description automatically generated**

1. Retrieve the title and type for each book in the book relation in which the type is SFI, HOR, ART or PSY. **Use the IN operator for this query.** Order the result by type ascending. Show the first five rows of the result. **(3 points)**

**SQL Statement:**

**Select title, type**

**FROM book**

**WHERE type IN (SFI, HOR, ART, PSY);**

**Output:** Table

Description automatically generated

1. Retrieve the title for each book in the book relation that begins with the word “The”. Order the result by title ascending. Show the full result. **(3 points)**

**SQL Statement:**

**SELECT \* FROM book**

**WHERE title LIKE 'The%';**

**Output:** **Table

Description automatically generated**

1. Retrieve the title for each book in the book relation that doesn’t have a type. Order the result by title ascending. Show the full result. **(3 points)**

**SQL Statement:**

**SELECT title**

**FROM book**

**WHERE type IS NULL**

**ORDER BY title ASC;**

**Output:** Table

Description automatically generated

1. Retrieve all of the columns from the author relation. Order the results by the authors lastName descending. **(3 points)**

**SQL Statement:**

**SELECT \* FROM author**

**ORDER BY lastName DESC;**

**Output:** **Table

Description automatically generated**

1. Retrieve the title and type from the book relation. Order the results first by type and then by title. Both sort keys should be ascending order. Show the first five rows of the result. **(3 points)**

**SQL Statement:**

**SELECT \* FROM book**

**ORDER BY type ASC, title;**

**Output:** **Table

Description automatically generated**

1. Retrieve a count of the number of books published by Penguin USA. Name the column ‘Penguin Books’. Order does not matter. Show your full result. **(3 points)**

**SQL Statement:**

**SELECT COUNT('Penguin Books')**

**FROM book;**

**Output:** **Graphical user interface, application, Teams

Description automatically generated**

1. Retrieve the publisherCode and the number of books from publisher from the book relation. Order the result by the count in descending order. Show the first five rows of the result. **(3 points)**

**SQL Statement:**

**SELECT publisherCode, SUM(books)**

**FROM book, publisher**

**ORDER BY COUNT(publisherCode) DESC;**

**Output:** Table

Description automatically generated**Table

Description automatically generated**

1. Retrieve the publisherCode and the number of books from that publisher from the book relation. Order the results by publisherCode and only show publishers who have 3 or more books in the relation. **Hint: will need to use the HAVING keyword.** Show your full result. **(3 points)**

**SQL Statement:**

**SELECT publisherCode, SUM(books(publisher))**

**FROM book, publisher**

**ORDER BY publisherCode >= 3;**

**Output:**

Table

Description automatically generated

Table

Description automatically generated

1. Retrieve the number of books in the book relation whose prices is $20.00 or lower. Order does not matter. Show your full result. **(3 points)**

**SQL Statement:**

**SELECT price**

**FROM book**

**WHERE price <= 20 ;**

**Output:** **Table

Description automatically generated**

1. Retrieve the title of the most expensive book in the book relation. Use a subquery for this problem. Show your full result. **(3 points)**

**SQL Statement:**

**SELECT price, title**

**FROM book**

**WHERE price = (**

**SELECT MAX(price)**

**FROM book);**

**Output:Table

Description automatically generated**

1. Retrieve the title of the least expensive book in the book relation. Use a subquery for this problem. Show your full result. **(3 points)**

**SQL Statement:**

**SELECT price, title**

**FROM book**

**WHERE price = (**

**SELECT MIN(price)**

**FROM book);**

**Output:** **Table

Description automatically generated**

1. Retrieve the title of all books in the book relation that are not published in New York. Order result by the city ascending. Use a subquery for this query. **(5 points)**

**SQL Statement:**

**SELECT title**

**FROM book**

**WHERE title = (SELECT title**

**WHERE title NOT IN ('New York'));**

**Output:** **Table

Description automatically generated**

1. Retrieve all of the columns from the book and publisher relations in one result. Use aliases in your query and use the simple JOIN syntax. Order does not matter. **(5 points)**

**SQL Statement:**

**SELECT b.publisherCode, bookCode, title, type, price, paperback, p.publisherCode, city**

**FROM book b JOIN publisher p;**

**Output:**

Table

Description automatically generated

1. Rewrite the previous query using the ON keyword. **(5 points)**

**SQL Statement:**

**SELECT b.publisherCode, bookCode, title, type, price, paperback, p.publisherCode, city**

**FROM book b JOIN publisher p ON(b.publisherCode = p.publisherCode);**

**Output:**

**Table

Description automatically generated**

1. Retrieve the title from the book relation and the city from the publisher relation using a JOIN query. Use aliases in your query. Order the result by title. **(5 points)**

**SQL Statement:**

**SELECT b.title, p.city**

**FROM book b JOIN publisher p;**

**Output:** **Table

Description automatically generated**

1. Retrieve the title from the book relation and the author lastName from the author relation. Order by author lastName. Use aliases in your query. **This will involve JOINING the book, author and wrote relations.** **(5 points).**

**\*\*\*Note, I noticed that the wrote relation on the assignment board does not have a lastName, but I included it a the question asks. It still runs the query fine.**

**SQL Statement:**

**SELECT t.title, l.lastName**

**FROM book t JOIN author l JOIN wrote**

**ORDER BY l.lastName;**

**Output:**

**Table

Description automatically generated**

1. Retrieve the title from the book relation and branchNum and onHand from the inventory relation. Use aliases in your query. Order the result by title. **(5 points)**

**SQL Statement:**

**SELECT t.title, i.branchNum, i.onHand**

**FROM book t JOIN inventory i**

**ORDER BY t.title;**

**Output:** **Table

Description automatically generated**

1. Retrieve the title from the book relation, the branchName from the branch relation and number of copies onHand from the inventory relation. Use aliases in your query. Order the result by title ascending. **(5 points)**

**SQL Statement:**

**SELECT t.title, b.branchName, i.onHand**

**FROM book t, branch b, inventory i**

**ORDER BY t.title ASC;**

**Output:** **Graphical user interface, application, table

Description automatically generated**

1. Retrieve the title from the book relation and compute the number of copies of the title that all branches have on hand. Name this computed column ‘Inventory’ **Hint: You will need to join book and inventory and do an aggregate query.** Use aliases in your query. Order the result by the total number of copies of the book in descending order. Show the first two rows of your result. **(5 points)**

**SQL Statement:**

**SELECT b.title, COUNT(i.onHand) AS Inventory**

**FROM book b JOIN inventory i**

**GROUP BY b.title**

**ORDER BY Inventory DESC;**

**Output: Table

Description automatically generated**

1. Retrieve the first name and last name from the author relation and the title from the book relation for all paperback books in the book relation. Order the result by the author last name and title. **(5 points)**

**SQL Statement:**

**SELECT a.firstName, a.lastName, b.title**

**FROM author a JOIN book b**

**ORDER BY a.firstName, b.title;**

**Output:** **Table

Description automatically generated**

1. Insert a new branch into the branch relation with the following data branch number = 5, branch name = Henry Lexington Green, branch location = 127 South Road, numEmployees = 7. Then write the query to show all of the branches in the branch relation. **(4 points)**

**SQL Statement:**

**SELECT \* FROM branch;**

**INSERT INTO branch(branchNum, branchName, branchLocation, numEmployees)**

**VALUES(('5'), ('Henry Lexington Green'), ('127 South Road'), ('7'));**

**Output:**

Table

Description automatically generated

1. The Henry Downtown branch moved to 184 St. John’s Way. Update the branch table with the new address. **(3 points)**

**SQL Statement:**

**SELECT \* FROM branch;**

**INSERT INTO branch(branchNum, branchName, branchLocation, numEmployees)**

**VALUES(('5'), ('Henry Lexington Green'), ('127 South Road'), ('7'));**

**UPDATE branch**

**SET branchLocation= '184 St. John’s Way'**

**Output:**

Table

Description automatically generated