# writing_evaluating_sql_queries

March 3, 2023

## 1 Writing and Evaluating SQL Queries

Tony Liang 39356993

CPSC 368

Mar 03, 2023

### 1.1 Part 1 Write SQL query to solve:

Template:

```
SELECT
FROM
WHERE
GROUP BY
HAVING
```

1. Find the Canadian authors who has never not had a bestseller (i.e., all their books are best sellers).

```
CREATE VIEW HadBestSeller(authorID) AS
SELECT w.authorID
FROM   Book b, Written w
WHERE  b.bestSeller = 1  AND w.bid = b.isbn

SELECT *
FROM    Author a NATURAL JOIN HadBestSeller hbs
WHERE   a.nationality = 'Canada'
```

2. Which genre(s) are the most popular amongst the books that have been on a bestseller list?

```
SELECT      bt.genreName
FROM        Book b BookType bt
WHERE       b.bestseller = 1
GROUP BY    bt. genreName
HAVING      Count(*) >=     ALL(SELECT Count(*)
                                FROM Book b2, BookType bt2
                                WHERE b2.bestseller = 1
                                GROUP BY bt2.genreName)
```

3. Find the first and last names of all authors who have written more than one book. Your answer must use an aggregate operator.

```
CREATE VIEW authorWrittenOne(id) AS
SELECT  a.id
FROM    Author a, Written w
WHERE   a.id = w.authorID
GROUP BY id
HAVING COUNT(a.id) = 1

SELECT  firstName, lastName
FROM    Author a NATURAL INNER JOIN Written w
WHERE   a.id NOT IN (SELECT * FROM authorWrittenOne)
```

4. Find the first and last names of all authors who have written more than one book. Your answer must not use an aggregate operator.

```
SELECT  firstName lastName
FROM    Author a NATURAL INNER JOIN Written w
WHERE   a.id NOT UNIQUE (   SELECT a.id FROM
                            FROM written w2
                            WHERE a.id = w2.authordID)
```

5. How many books has each province borrowed?

```
SELECT      COUNT(Distinct b.isbn), a.province
FROM        Address a, Borrows br, User u, Book b
WHERE       a.id = u.addressID AND
            br.userID = u.id AND
            br.bid = b.isbn
GROUP BY    a.province
```

6. Each province/territory has run a campaign to encourage residents to use the library more often. Find the provinces and/or territories that have had an increase in the number of people who have borrowed a book in 2022 when compared to their average across previous years

```
CREATE VIEW countBefore2022(counts) AS
SELECT      COUNT(DISCTINCT b.bid, b.userID)
FROM        Borrows br, Book b, User u
WHERE       b.borrowYear < 2022 AND br.bid = b.isbn AND
            br.userID = u.id
GROUP BY    b.borrowYear

SELECT      province
FROM        Address a NATURAL JOIN Borrows br NATURAL JOIN User u
WHERE       borrowYear = 2022
GROUP BY    a.province
HAVING      COUNT( DISTINCT b.bid, b.userID) > AVG(SELECT * countBefore2022)
```

7. Find the most common genre for each author.

```
CREATE VIEW countGenres(count) AS
```

```
SELECT COUNT(*)
FROM Author NATURAL JOIN Written NATURAL JOIN BookType
GROUP BY genreName

SELECT      a.id, bt.genreName
FROM        Author a NATURAL JOIN BookType bt
GROUP BY    a.id, bt.genreName
HAVING      COUNT(*) > ALL (SELECT * FROM countGenres)
```

8. Let's assume that we have divided our library users into the following age groups: young readers (ages 0 to 12), young adults (13 to 19), adults (19 to 55), and seniors (55+). Find the books that have been read by every age group.

```
CREATE VIEW notReadBooks(isbn, title) AS
SELECT      b.isbn, DISTINCT b.title
FROM        Book b, Borrows br, User u
WHERE       b.id = br.bid AND br.userID = u.id AND
GROUP BY    b.isbn, b.title
HAVING      COUNT(*) = 0

SELECT      b.title
FROM        Book b
WHERE       b.isbn NOT IN (SELECT * FROM notREADBooks)
```

9. Find the number of brand new users for libraries in each province and territory for 2022.

```
CREATE VIEW oldUsers(id) AS
SELECT      u.id
FROM        Address a, Borrows br, User u
WHERE       a.id = u.id AND br.year < 2022 AND br.userID = u.id

SELECT      COUNT(*)
FROM        Address a, Borrows br, User u
WHERE       a.id = u.id AND
            br.borrowYear = 2022 AND
            br.userID = u.id AND
            u.id NOT IN (oldUsers.id)
GROUP BY    a.province
```

10. Find the most popular genre for each year.

```
CREATE VIEW countGenres(count) AS
SELECT      COUNT(*)
FROM        BookType bt NATURAL JOIN Borrows br NATURAL JOIN User u
GROUP BY    genreName

SELECT      br.borrowYear, bt.genreName
FROM        BookType bt NATURAL JOIN Borrows br NATURAL JOIN User u
GROUP BY    br.borrowYear, bt.genreName
HAVING      COUNT(*) >= ALL (SELECT * FROM countGenres)
```

11. Find the books that have been read in every province.

```
// This one is very weird
SELECT      DISTINCT b.title
FROM        Book b, Borrows br, User u NATURAL JOIN Address a
WHERE       b.isbn = br.bid AND br.userID = u.id
GROUP BY    a.province
HAVING      COUNT(DISTINCT b.title) = COUNT(u.id)
```

12. Find the most popular book borrowed for each province each year.

```
CREATE VIEW bookCounts(userID) AS
SELECT      COUNT(br.userID)
FROM        Book b, Borrows br, User u, Address a
WHERE       u.addressID = a.id AND
            br.userID = u.id AND
            br.bid = b.isbn
GROUP BY    a.province, br.borrowYear

SELECT      b.title, a.province, a.borrowYear
FROM        Borrows br Natural JOIN Book b NATURAL JOIN User u NATURAL JOIN Address a
GROUP BY    a.province, br.borrowYear
HAVING      COUNT(br.userID) > ALL(SELECT * FROM bookCounts bc)
```

13. For each province/territory, find the year it had the highest number of users.

```
CREATE VIEW bookCounts(userID) AS
SELECT      COUNT(br.userID)
FROM        Book b, Borrows br, User u, Address a
WHERE       u.addressID = a.id AND
            br.userID = u.id AND
            br.bid = b.isbn
GROUP BY    br.borrowYear


SELECT      br.borrowYear, a.province
FROM        Address a, Borrows br, User u
WHERE       u.id = br.userID AND
            br.bid = b.isbn AND
            u.addressID = a.id
GROUP BY    br.borrowYear, a.province
HAVING      COUNT(br.userID) > ALL(SELECT * FROM bookCounts bc)
```

14. Find the nationality that has had the most bestsellers.

```
CREATE VIEW         bestSellersPerCountry bspc AS
SELECT              COUNT(*)
FROM                Author a NATURAL JOIN Book b NATURAL JOIN Written w
WHERE               b.bestSeller = 1
GROUP BY            a.id
```

```
SELECT              a.nationality, a.id
FROM                Author a NATURAL JOIN Book b NATURAL JOIN Written w
WHERE               b.bestSeller = 1
GROUP BY            a.id
HAVING COUNT(*)     >= ALL(SELECT COUNT(*) FROM bestSellersPerCountry bspc)
```

15. Find the youngest person who has written a bestseller.

```
CREATE VIEW dobBestSeller(yearOfBirth) AS
SELECT  a.yearOfBirth
FROM    Author a NATURAL JOIN Written w NATURAL JOIN Book b
WHERE   b.bestSeller = 1

SELECT  *
FROM    Author a NATURAL JOIN Written w NATURAL JOIN Book b
WHERE   b.bestSeller = 1 AND a.yearOfBirth = (SELECT MAX(yearOfBirth) FROM dobBestSeller d
```

16. Find the people who have borrowed the same book multiple times.

```
SELECT      u.id, DISTINCT u.firstName, u.lastName
FROM        Borrows br NATURAL JOIN User u
GROUP BY    b.bid, b.userID
HAVING      COUNT(*) > 1
```

BACK TO TOP

### 1.2 Part 2 Evaluate SQL query

## 2 Part 2 Evaluate SQL query

1. Find all books which have never been borrowed.

```
SELECT DISTINCT b1.isbn
FROM Book b1, Borrows b2
WHERE b1.isbn = b2.bid AND b1.isbn <> b2.bid
```

I believe this is not correct, an example below could prove this:

If we had isbn in {1,2,3}, bid of {1,2}. If the query above is ran, then yields to a following result:

| isbn | bid |
|---|---|
| 1 | 2 |
| 2 | 1 |
| 3 | 1 |
| 3 | 2 |

However, we did not have isbn 3 borrowed in the first place, while it is included in the select statement.