4b. Structured Query Language
Exercises 3
Find the number of sailor records
SELECT  **COUNT**  (*)
FROM   Sailors

Find the average age of sailors with rating 10
SELECT  **AVG** (S.age)
FROM   Sailors as S
WHERE  S.rating=10

Find how may different ratings exist for sailors who are called Bob
SELECT  **COUNT**  (DISTINCT S.rating)
FROM   Sailors as S
WHERE S.sname='Bob'
- Find name and age of the oldest sailor(s)

```
SELECT  S.sname, MAX (S.age)
FROM   Sailors as S
```

Illegal! If there is no group by, we can only have aggregates (without attributes). The correct solution is:

```
SELECT  S.sname, S.age
FROM   Sailors as S
WHERE  S.age =
       (SELECT  MAX (S2.age)
        FROM   Sailors as S2)
```
Sailors (sid, sname, rating, age),
Reserves (sid, bid, date),
Boats (bid, bname, color)

For each red boat, display the bid and the number of reservations for this boat
SELECT  B.bid, COUNT (*) AS scount
FROM  Boats as B, Reserves as R
WHERE R.bid=B.bid AND B.color=’red’
GROUP BY  B.bid

For each red boat, display the name and the number of reservations for this boat
SELECT  B.bname, COUNT (*) AS scount
FROM  Boats as B, Reserves as R
WHERE R.bid=B.bid AND B.color=’red’
GROUP BY  B.bid, B.bname
Find the age of the youngest sailor with age > 18, for each rating with at least 2 such sailors.

SELECT S.rating, MIN (S.age)
FROM Sailors as S
WHERE S.age > 18
GROUP BY S.rating
HAVING COUNT (*) > 1

Answer relation Sailors (sid, sname, rating, age),
Reserves (sid, bid, date),
Boats (bid, bname, color)
• Find the age of the youngest sailor with age > 18, for each rating with at least 2 sailors (of any age)

```sql
SELECT  S.rating,  MIN (S.age)
FROM  Sailors as S
WHERE  S.age > 18
GROUP BY  S.rating
HAVING  1  <  (SELECT  COUNT (*)
FROM  Sailors as S2
WHERE  S.rating=S2.rating)
```
Find those ratings for which the average age is the minimum over all ratings

```
SELECT Temp.rating, Temp.avgage
FROM (SELECT S.rating, AVG (S.age) as avgage
    FROM Sailors as S
    GROUP BY S.rating) as Temp
WHERE Temp.avgage = (SELECT MIN (Temp.avgage)
    FROM Temp)
```
SQL-Algebra Review

• Suppose a bookstore has the following five relational tables:
  BOOK (BID, TITLE, AID, SUBJECT, QUANTITY-IN-STOCK)
  AUTHOR (AID, FIRST-NAME, LAST-NAME)
  CUSTOMER (CID, FIRST-NAME, LAST-NAME)
  ORDER-DETAILS (OID, BID, QUANTITY)
  ORDER (OID, CID, ORDER-YEAR)

• ASSUMPTIONS:
  Keys are underlined and foreign keys are in italics. Each author has authored at least one book in the store. Each book has exactly one author. Each order is made by exactly one customer and has one or more associated record in ORDER-DETAILS (e.g., an order may contain different books).
Write the following queries in relational algebra and SQL.

**Query 1:** Find all distinct book titles of the author whose last name is “Luo”.

**algebra:**

\[
\pi_{\text{TITLE}} (\sigma_{\text{LAST-NAME} = 'Luo'} (\text{BOOK JOIN} \quad \text{AUTHOR.AID = BOOK.AID} \quad \text{AUTHOR}))
\]

**SQL:**

```
SELECT DISTINCT B.TITLE
FROM BOOK AS B, AUTHOR AS A
WHERE A.LAST-NAME = 'Luo' AND A.AID = B.AID
```
Query 2: Find the last name and first name of all authors who wrote books in both the subjects of “ART” and “BUSINESS”.

**algebra:**

\[
\pi_{\text{LAST-NAME,FIRST-NAME}}
( ( \pi_{\text{AID, LAST-NAME, FIRST-NAME}} ( \sigma_{\text{BOOK.SUBJECT = 'ART'}} (\text{BOOK JOIN}\ \text{AUTHOR.AID = BOOK.AID}\ \text{AUTHOR}))) \cap
( \pi_{\text{AID, LAST-NAME, FIRST-NAME}} ( \sigma_{\text{BOOK.SUBJECT = 'BUSINESS'}} (\text{BOOK JOIN}\ \text{AUTHOR.AID = BOOK.AID}\ \text{AUTHOR}))))
\]

**SQL:**

\[
\text{SELECT A1.LASTNAME, A1.FIRSTNAME}
\text{FROM AUTHOR AS A1, BOOK AS B1}
\text{WHERE A1.AID = B1.AID AND B1.SUBJECT = 'ART' AND A1.AID IN}
\text{(SELECT A2.AID}
\text{FROM AUTHOR AS A2, BOOK AS B2}
\text{WHERE A2.AID = B2.AID AND B2.SUBJECT = 'BUSINESS'})
\]
• **Query 3:** Find the IDs of customers that have ordered all books in the subject of “ART”.

  • **algebra:**

\[
\left( \pi_{CID, BID} \right) \left( \text{ORDERDETAILS JOIN}_{\text{ORDER}.OID=\text{ORDERDETAILS}.OID} \text{ORDER} \right)
\]

\[
/ \left( \pi_{BID} \left( \sigma_{\text{BOOK}.SUBJECT = 'ART'} \text{BOOK} \right) \right)
\]

• **SQL:**

```
SELECT CID
FROM CUSTOMER
WHERE NOT EXISTS

\left( \text{SELECT BID}
\text{FROM BOOK}
\text{WHERE SUBJECT = 'ART'} \right)
\text{EXCEPT}
\left( \text{SELECT ORDERDETAILS}.BID
\text{FROM ORDER, ORDERDETAILS}
\text{WHERE ORDER}.CID=\text{CUSTOMER}.CID \text{ AND ORDER}.OID=\text{ORDERDETAILS}.OID \right)
```

---

BOOK (BID, TITLE, AID, SUBJECT, QUANTITY-IN-STOCK),
AUTHOR (AID, FIRST-NAME, LAST-NAME),
CUSTOMER (CID, FIRST-NAME, LAST-NAME), ORDER-DETAILS (OID, BID, QUANTITY), ORDER (OID, CID, ORDER-YEAR)
• **Query 4**: Find the last name and first name of all authors who wrote books in at least two subjects.

• **algebra:**

\[
\pi_{\text{LAST-NAME, FIRST-NAME}} (\sigma_{\text{B1.SUBJECT} \neq \text{B2.SUBJECT}}
\left(\rho(\text{B1,BOOK}) \Join_{\text{AID}} \text{AUTHOR} \Join_{\text{AID}} \rho(\text{B2,BOOK}))\right))
\]

• **SQL:**

```sql
SELECT A.LASTNAME, A.FIRSTNAME
FROM AUTHOR A, BOOK B1, BOOK B2
WHERE B1.AID=A.AID AND B2.AID=A.AID AND
B1.SUBJECT \neq B2.SUBJECT
```
For the following queries only use SQL

Query 5: Find the last name and first name of all authors who wrote books in exactly two subjects.

SQL:
SELECT A.LASTNAME, A.FIRSTNAME
FROM AUTHOR A, BOOK B
WHERE B.AID=A.AID
GROUP-BY A.AID, A.LASTNAME, A.FIRSTNAME
HAVING COUNT(DISTINCT SUBJECT)=2
• **Query 6:** For each customer, display the CID, last-name, and the number of orders in 2010 - include only customers that made more than 100 orders in 2010.

• **SQL:**

```sql
SELECT CUSTOMER.CID, LAST-NAME, COUNT(*)
FROM ORDER, CUSTOMER
WHERE ORDER.CID=CUSTOMER.CID AND ORDER-YEAR=2010
GROUP-BY CUSTOMER.CID, LAST-NAME
HAVING COUNT(*)>100
```
• **Query 7**: Display the CID and last-name of customer(s) who have ordered the **largest total** quantity of books.

• **SQL**:

```sql
SELECT TEMP.CID, TEMP.LAST-NAME
FROM

(SELECT C.CID, LAST-NAME, SUM(QUANTITY) AS TOTALQ
FROM CUSTOMER C, ORDER O, ORDER-DETAILS OD
WHERE C.CID=O.CID AND O.OID = OD.OID
GROUP-BY C.CID, C.LAST-NAME) AS TEMP
WHERE TEMP.TOTALQ =

(SELECT MAX(TOTALQ) FROM TEMP)
```