SQL - 3

Week 7 - 2
Aggregate operators

• What is aggregation?
  – Computing arithmetic expressions, such as Minimum or Maximum

• The aggregate operators supported by SQL are: COUNT, SUM, AVG, MIN, MAX
Aggregate Operators

- **COUNT(A):** The number of values in the column A
- **SUM(A):** The sum of all values in column A
- **AVG(A):** The average of all values in column A
- **MAX(A):** The maximum value in column A
- **MIN(A):** The minimum value in column A

(We can use DISTINCT with COUNT, SUM and AVG to compute only over non-duplicated columns)
Using the COUNT operator

Count the number of sailors

```
SELECT COUNT (*)
FROM  Sailors S;
```
Another Aggregate Query

Count the number of different sailor names

SELECT COUNT (DISTINCT S.sname) FROM Sailors S;
Example of SUM operator

*Find the sum of ages of all sailors with a rating of 10*

```sql
SELECT SUM (S.age)
FROM Sailors S
WHERE S.rating=10;
```
Example of AVG operator

Find the average age of all sailors with rating 10

SELECT  AVG (S.age)
FROM    Sailors S
WHERE   S.rating=10;
Example of MAX operator

*Find the name and age of the oldest sailor*

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

But this is illegal in SQL!!
Correct SQL Query for MAX

SELECT S.sname, S.age
FROM Sailors S
WHERE S.age = ( SELECT MAX(S2.age)
               FROM Sailors S2 );
Alternatively…

```sql
SELECT S.sname, S.age
FROM Sailors S
WHERE ROWNUM <= 1
ORDER BY S.age DESC;
```
Banking Examples

branch (branch-id, branch-city, assets)

customer (customer-id, customer-name, customer-city)

account (account-number, branch-id, balance)

loan (loan-number, branch-id, amount)

depositor (customer-id, account-number)

borrower (customer-id, loan-number)
IN...Example 1

“Find the account numbers opened at branches of the bank in Fairfax”

SELECT A.account-number
FROM account A
WHERE A.branch-id IN (SELECT B.branch-id
                        FROM branch B
                        WHERE B.branch-city= ‘Fairfax’ )
IN...Example 2

“Find the account numbers opened at branches 101 and 102 of the bank”

SELECT A.account-number
FROM A.account
WHERE A.branch-id IN (‘101’, ‘102’)
The *EXISTS* predicate is TRUE if and only if the Subquery returns a non-empty set.

The *NOT EXISTS* predicate is TRUE if and only if the Subquery returns an empty set.

The *NOT EXISTS* can be used to implement the SET DIFFERENCE operator from relational algebra.
EXISTS...Example 1

“Select all the account balances where the account has been opened in a branch in Fairfax”

SELECT A.account-balance
FROM account A
WHERE EXISTS (SELECT *
FROM branch B
WHERE B.branch-city=‘Fairfax’
AND B.branch-id=A.branch-id)
**EXISTS** … Example 2

“Select all the account balances where the account has not been opened in a Fairfax branch”

```
SELECT A.account-balance
FROM account A
WHERE NOT EXISTS (SELECT *
    FROM branch B
    WHERE B.branch-city=‘Fairfax’
    AND B.branch-id=A.branch-id)
```
Quantified Comparison Predicate
Example 1

“Select account numbers of the accounts with the minimum balance”

SELECT A.account-number
FROM account A
WHERE A.balance <= ALL (SELECT A2.balance
FROM account A2)
Aggregate Functions in SQL… revisited

SQL provides five built-in aggregate functions that operate on sets of column values in tables:

\[ \text{COUNT(), MAX(), MIN(), SUM(), AVG().} \]

With the exception of \( \text{COUNT()} \), these set functions must operate on sets that consist of simple values—that is, sets of numbers or sets of character strings, rather than sets of rows with multiple values.
Aggregate Functions in SQL

Example 1

“Select the total amount of balance of the account in branches located in Fairfax”

SELECT SUM(A.balance) AS total_amount
FROM account A, branch B
WHERE B.branch-city=‘Fairfax’ AND
    B.branch-id= A.branch-id
Aggregate Functions in SQL
Example 2

“Select the total number of opened accounts”

SELECT COUNT(A.account-number)
FROM account A

OR

SELECT COUNT(*)
FROM account