SQL Overview

•SQL is the language of relational databases.
•It is used for every aspect of database development and management.

Objectives of SQL

•Ideally, database language should allow user to:
-create the database and relation structures;
-perform insertion, modification, deletion of data from relations;
-perform simple and complex queries.
•Must perform these tasks with minimal user effort and command structure/syntax must be easy to learn.
•It must be portable.

Importance of SQL

•SQL has become part of application architectures such as IBM's Systems Application Architecture.
•It is strategic choice of many large and influential organizations (e.g. X/OPEN).
•SQL is Federal Information Processing Standard (FIPS) to which conformance is required for all sales of databases to American Government.

Nature of SQL

•SQL is a declarative language.
•Procedural languages like C# or Java describe how to accomplish a task step by step.
•In a declarative language, you say *what* you want to do, not *how*.

SQL Functionality

•SQL is not case sensitive.
•In some environments SQL statements must be ended with a semicolon.
•SQL is usually divided into two broad areas of functionality:
-DDL (Data Definition Language): Data definition language is the set of SQL keywords and commands used to create, alter, and remove database objects.
-DML (Data Manipulation Language): Data manipulation language is the set of key words and commands used to retrieve and modify data. SELECT, UPDATE, INSERT, and DELETE are the primary actions of DML.

Writing SQL Commands

•SQL statement consists of *reserved words* and *user-defined words*.
–Reserved words are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.
–User-defined words are made up by user and represent names of various database objects such as relations, columns, views.
•Most components of an SQL statement are *case insensitive*, except for literal character data.
•More readable with indentation and lineation:

•Each clause should begin on a new line.
•Start of a clause should line up with start of other clauses.
•If clause has several parts, should each appear on a separate line and be indented under start of clause.
•Use extended form of BNF notation:
- Upper-case letters represent reserved words.
- Lower-case letters represent user-defined words.
- | indicates a *choice* among alternatives.
- Curly braces indicate a *required element*.
- Square brackets indicate an *optional element*.
- … indicates optional repetition (0 or more).

Literals

•Literals are constants used in SQL statements.
•All non-numeric literals must be enclosed in single quotes (e.g. 'London').
•All numeric literals must not be enclosed in quotes (e.g. 650.00).

Select Statement

•The SELECT statement is used to retrieve data from the database.
•The basic syntax is:

```
SELECT <columnName>, <columnName>
FROM <TableName>
```

```
SELECT StudentFirstName, StudentLastName, StudentPhone
FROM Student
```

List all staff with a salary greater than 10,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > 10000;
```

List addresses of all branch offices in London or Glasgow.

```
SELECT *
FROM Branch
WHERE city = 'London' OR city = 'Glasgow';
```

List all staff with a salary between 20,000 and 30,000.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary BETWEEN 20000 AND 30000;
```

•BETWEEN test includes the endpoints of range.

The * WildCard
•Instead of listing each of the columns, you can use an * to include all columns.

```
  SELECT * FROM
```

•Listing the columns does give you the ability to choose both which columns and which order to present them.
•With the * you return all the columns in the order they have in the underlying table.

Distinct Key Word
•Sometimes a query will return multiple duplicate values.
•For instance the statement
SELECT Key
FROM Session
Could return numerous instances of each customer.
•The DISTINCT keyword will make it so it only returns one instance of each Key.
SELECT DISTINCT Key
FROM Session
•The DISTINCT keyword always operates on the whole role, not on individual columns.
•It only returns distinct rows.

Calculations
•You can do calculations in SELECT statements.
SELECT ItemNumber, ItemPrice, Quantity, **ItemPrice * Quantity**
FROM CustomerOrder

Order of Operations
The order of operation is the same as in algebra.
1.Whatever is in parentheses is executed first. If parentheses are nested, the innermost is executed first, then the next most inner, etc.
2.Then all division and multiplication left to right
3.And finally all addition and subtraction left to right

Sorting
•You can sort the results of a query by using the keywords ORDER BY.
SELECT *
FROM Session
ORDER BY SessionDate
•ORDER BY does an ascending A-Z, 1-10, etc. sort by default.
•You can change the direction by using the DESC keyword after the field to be sorted.

Aliasing
•Sometimes it is useful to alias a column name to make a more readable result set.
SELECT StudentLastName AS [Last Name], StudentFirstName AS [First Name]
FROM Student
•The AS keyword is optional.
•Double quotes “ “ can be used instead of square brackets.

Where Clause
•The WHERE clause allows you to limit the rows you return in a query.
•You use the WHERE clause to specify the criteria by which the rows will be filtered.
SELECT LastName, FirstName, Phone, City
FROM Customer
WHERE City = ‘Boston’

Like
•The LIKE keyword used in a WHERE operator with a wildcard (% or _) allows you to search for patterns in character-based fields.
•The following returns all items whose name starts with “T.”
SELECT ItemName, ItemPrice
FROM Inventory

WHERE ItemName LIKE ’T%’

Between
•The BETWEEN keyword can be used in criteria to return values between to other values.
•BETWEEN is inclusive of its ends.
SELECT Key, SessionDate, StudentKey
FROM Session
WHERE SessionDate BETWEEN ’11/1/2014’ AND ’11/30/2014’

AND OR NOT
•You can use keywords AND, OR, and NOT to combine criteria in a query.
•AND is exclusive. Or is Inclusive.
•WHERE City = ‘Boston’ OR City=‘Los Angeles’ returns all records that have either Boston or Los Angeles for their city.
•WHERE City=‘Boston’ AND City=‘Los Angeles’ returns nothing because the record cannot have both at the same time.
•NOT excludes.
•WHERE NOT City = ‘Los Angeles’ returns every city except Los Angeles.

NULL
•Nulls are special cases. They are not a value and so cannot be compared to a value using = or < or >.
•To locate nulls you can use the IS keyword in a criteria:
WHERE StudentKey IS NULL
WHERE StudentKey IS NOT NULL

Select Set Membership
•There is a negated version (NOT IN).
• IN does not add much to SQL’s expressive power. Could have expressed this as:
```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position=‘Manager’ OR
          position=‘Supervisor’;
```
• IN is more efficient when set contains many values.

Select Pattern Matching
Find all owners with the string ‘Glasgow’ in their address.
SELECT ownerNo, fName, lName, address, telNo
FROM PrivateOwner
WHERE address LIKE ‘%Glasgow%’;
•SQL has two special pattern matching symbols:
•%: sequence of zero or more characters;
•_ (underscore): any single character.
•LIKE ‘**%Glasgow%**’ means a sequence of characters of any length containing ‘*Glasgow*’.

Select NULL Search Condition
List details of all viewings on property PG4 where a comment has not been supplied.
•There are 2 viewings for property PG4, one with and one without a comment.
•Have to test for null explicitly using special keyword IS NULL:
```
SELECT clientNo, viewDate
FROM Viewing
WHERE propertyNo = ‘PG4’ AND comment IS NULL;
```