# Retrieving Data Using the SQL SELECT Statement

Practical 4

# Lesson Objectives

- Learn the capabilities of SQL SELECT statement.
- Execute a basic SQL SELECT statement.

# Introduction to SQL

- Structured Query Language (SQL): The standard query language for relational databases.
  - **Data Query Language (DQL)**
    - **View database data – Select.**
  - Data Manipulation Language (DML)
    - Insert, update, delete, merge database data.
  - Data Definition Language (DDL)
    - Create new database objects.
    - Modify or delete existing database objects.
  - Data Control Language (DCL)
    - Grant or revoke privileges and assign storage area to user.
  - Transaction Control Language (TCL)
    - Statement used to manage the changes made by DML.
    - COMMIT, ROLLBACK, SAVEPOINT.

## Using SQL Plus

- Run the SQL Plus
  - Username : system
  - Password   : oracle

## Using Scripts

- One or more SQL commands can be saved in a text file.
- The text file usually have .sql extension.
- To run the text file from SQL*Plus:
  - **start C:\myfile.sql**

    *OR*
  - **@C:\myfile.sql**
  - The extension can be omitted if it is .sql

## DESCRIBE Command

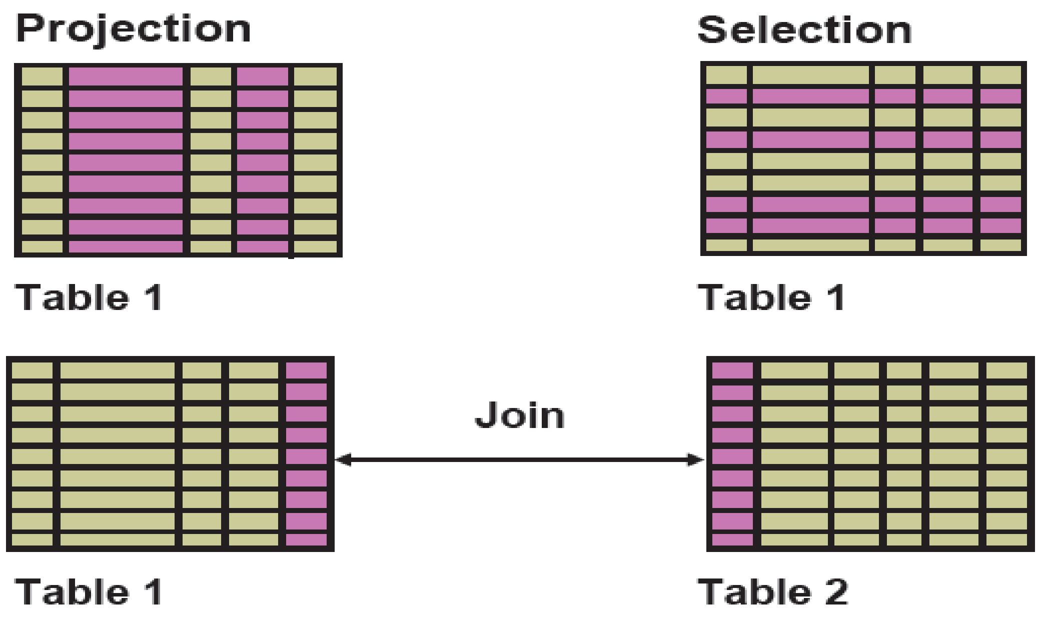
- To display the table structure:

**DESCRIBE** student

*OR*

**DESC** student

## Capabilities of SQL SELECT Statements



## Basic SELECT Statement

*SELECT *| { [DISTINCT ] column | expression [alias], ... }*
*FROM table;*

- SELECT identifies the columns to be displayed.
- FROM identifies the table containing those columns.

## SELECT All Columns

- Retrieve every record and field from the LOCATION table:

**SELECT ***

**FROM** location;

## SELECT Specific Columns

- Retrieve the student first name, middle initial, and last name from every row in the STUDENT table:

  **SELECT** s_first, s_mi, s_last

  **FROM** student;

## SELECT (Suppress Duplicate)

- Retrieve all faculty ranks from the FACULTY table:

  SELECT f_rank

  FROM faculty;

- To retrieve and suppress duplicate rows:

  SELECT **DISTINCT** f_rank

  FROM faculty;

## Writing SQL Statements

- SQL statements are not case sensitive.
- SQL statements can be entered on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- In SQL*Plus, you are required to end each SQL statement with a semicolon (;).

## Creating Search Conditions in SQL Queries

- An expression that seeks to match specific table records.
- Used in SELECT, UPDATE and DELETE statements.
- *WHERE fieldname comparison_operator search_expression*

## Defining Search Expressions

- Character Strings
  - Must be enclosed in single quotes.
  - It is case sensitive.

```
SELECT s_last, s_first, s_dob
FROM student
WHERE s_first = 'Sarah';

SELECT s_last, s_first, s_dob
FROM student
WHERE s_first = 'SARAH';
```

## Exact Search Condition

- An exact search condition uses the equal to comparison operator (**=**) to match a value exactly.

```
SELECT f_first, f_mi, f_last, f_rank
FROM faculty
WHERE f_rank = 'ASSO';
```

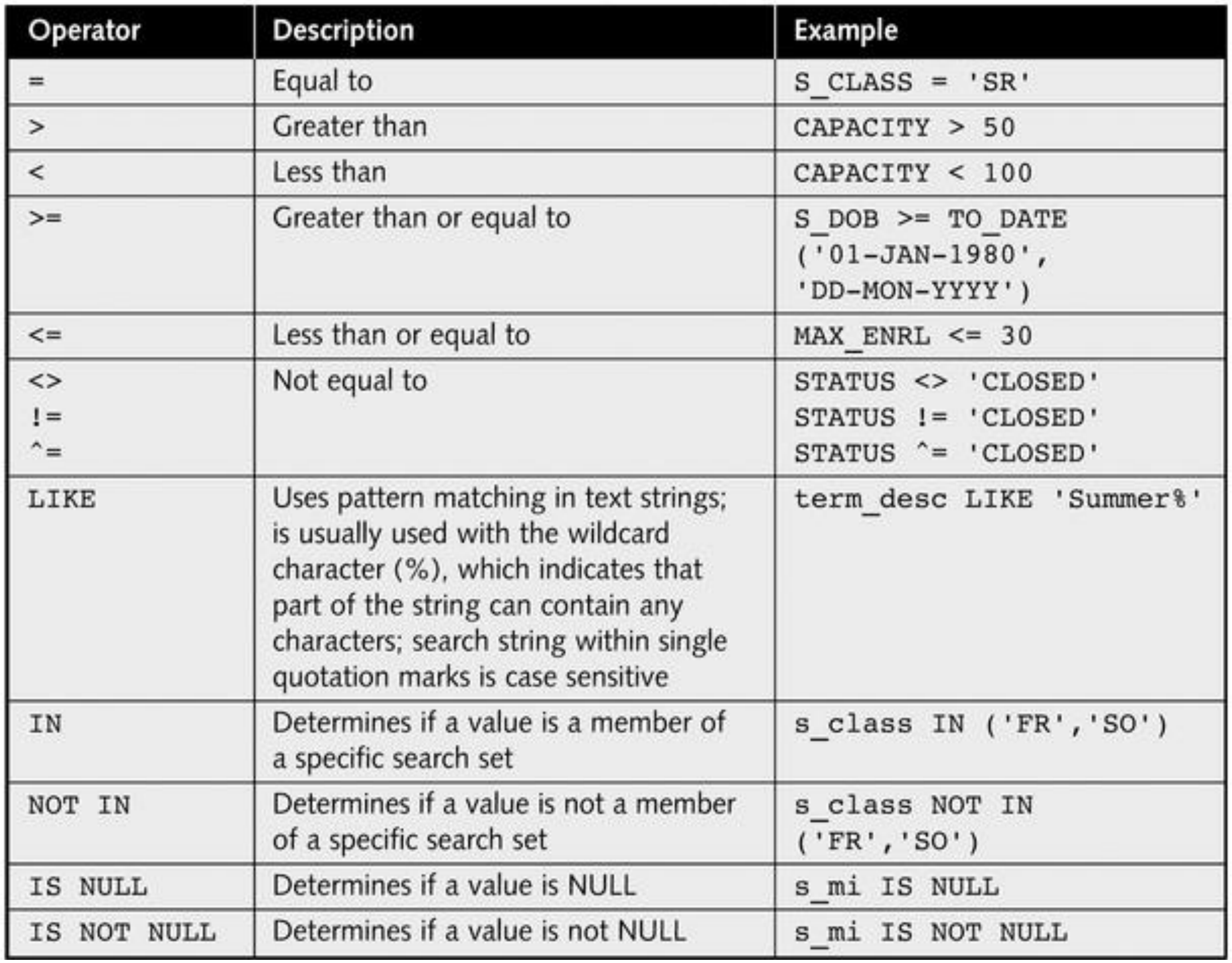
## Inexact Search Condition

- An inexact search condition uses the inequality comparison operators (>,<,>=,<=) to match a range of values.

```
SELECT bldg_code, room, capacity
FROM location
WHERE capacity >= 40;
```

# Comparison Operators

**Table 3-3** Common search condition comparison operators



| Operator | Description | Example |
|---|---|---|
| `=` | Equal to | `S_CLASS = 'SR'` |
| `>` | Greater than | `CAPACITY > 50` |
| `<` | Less than | `CAPACITY < 100` |
| `>=` | Greater than or equal to | `S_DOB >= TO_DATE ('01-JAN-1980', 'DD-MON-YYYY')` |
| `<=` | Less than or equal to | `MAX_ENRL <= 30` |
| `<>`<br>`!=`<br>`^=` | Not equal to | `STATUS <> 'CLOSED'`<br>`STATUS != 'CLOSED'`<br>`STATUS ^= 'CLOSED'` |
| `LIKE` | Uses pattern matching in text strings; is usually used with the wildcard character (%), which indicates that part of the string can contain any characters; search string within single quotation marks is case sensitive | `term_desc LIKE 'Summer%'` |
| `IN` | Determines if a value is a member of a specific search set | `s_class IN ('FR','SO')` |
| `NOT IN` | Determines if a value is not a member of a specific search set | `s_class NOT IN ('FR','SO')` |
| `IS NULL` | Determines if a value is NULL | `s_mi IS NULL` |
| `IS NOT NULL` | Determines if a value is not NULL | `s_mi IS NOT NULL` |

## Creating Complex Search Conditions

- Combines multiple search conditions using the AND, OR, and NOT logical operators.
- AND – both conditions must be true.
- OR – one or both condition must be true.
- NOT – opposite of actual value.
- Use () to group logical operators.

# Logical Operators

```
SELECT bldg_code, room, capacity
FROM location
WHERE bldg_code = 'BUS' AND capacity >= 40;

SELECT bldg_code, room, capacity
FROM location
WHERE bldg_code = 'BUS' OR capacity >= 40;

SELECT *
FROM student
WHERE NOT (s_class = 'FR');
```

## Range Conditions Using BETWEEN Operator

```
SELECT  s_id, s_last, s_zip
FROM student
WHERE s_zip BETWEEN 54701 AND 54705;

SELECT  s_id, s_last, s_first
FROM student
WHERE s_last BETWEEN 'Black' AND 'Mobley';
```

## NULL and NOT NULL Values

SELECT *
FROM enrollment
WHERE grade **IS NULL**;

SELECT *
FROM enrollment
WHERE grade **IS NOT NULL**;

## IN and NOT IN Comparison Operators

SELECT *
FROM enrollment
WHERE grade **IN** ('A', 'B');

SELECT *
FROM enrollment
WHERE grade **NOT IN** ('A', 'B');

## Practice 4.1

- Using comparison operator and logical operator to rewrite the following statement:

```
SELECT *
FROM enrollment
WHERE grade IN ('A', 'B');
```

# LIKE Comparison Operator

```
SELECT *
FROM term
WHERE term_desc LIKE '%2006';

SELECT *
FROM term
WHERE term_desc LIKE 'Fall%';

SELECT call_id
FROM course
WHERE call_id LIKE '%1__';
```

## Practice 4.2

- write a query to list all courses which contain "system" in its name.

## Practice 4.3

```
SELECT last_name
FROM emp
WHERE last_name LIKE '_o%';
```

Which of the following last names could have been returned from the above query?

1. Sommersmith
2. Kog
3. Fong
4. Mo

## Sorting Query Output

- You can sort query output by using the ORDER BY clause and specifying the sort key.
- The default sorting order is ascending, use DESC to sort the records in descending order.

## Sorting Query Output

```
SELECT bldg_code, room, capacity
FROM location
WHERE capacity >= 40;

SELECT bldg_code, room, capacity
FROM location
WHERE capacity >= 40
ORDER BY capacity;
```

## Sorting Query Output

```
SELECT bldg_code, room, capacity
FROM location
WHERE capacity >= 40
ORDER BY 2;

SELECT bldg_code, room, capacity
FROM location
WHERE capacity >= 40
ORDER BY capacity DESC;
```

## Sorting Query Output

- If the null ordering is not specified then the handling of the null values is:
  - NULLS LAST if the sort is ASC
  - NULLS FIRST if the sort is DESC

```
SELECT  *
FROM enrollment;
```

```
SELECT  *
FROM enrollment
ORDER BY grade NULLS FIRST;
```

```
SELECT  *
FROM enrollment
ORDER BY grade;
```

```
SELECT  *
FROM enrollment
ORDER BY grade NULLS LAST;
```

## Practice 4.4

- Display the  bldg_code, room, capacity in which the capacity is greater than or equal to 35 seats in BUS and CR building, sort the list by bldg_code in descending order and room in ascending order.

## Using Calculations in SQL Queries

- Calculations are performed using the arithmetic operators (**+**, **-**, *****, **/**).
- Calculations can be performed on NUMBER, DATE and INTERVAL fields only.

## Arithmetic with Dates

| Operation | Result |
|---|---|
| date + number | Date |
| date – number | Date |
| date – date | Number of days |
| date + number/24 | Date |

## SYSDATE Function

- SYSDATE is a date function that returns the current database server date and time.

## Using Arithmetic Operators

```
SELECT course_id, course_name, credits * 100
FROM course;

SELECT s_id, s_last, (SYSDATE-s_dob)/365.25
FROM student;
```

## Operator Precedence

```
SELECT bldg_code, room, capacity
FROM location;

SELECT bldg_code, room, capacity + 10
FROM location;

SELECT bldg_code, room, capacity + 10 * 2
FROM location;
```

## Using Parentheses

- You can override the rules of precedence by using parentheses to specify the desired order in which the operators are to be executed.

```
SELECT bldg_code, room,
(capacity + 10) * 2
FROM location;
```

## Column Alias

- Use an alias for column headings:
  *SELECT fieldname1 AS alias_name1* …
- Requires double quotation marks if it contains space or special characters (# or $), or it is case-sensitive.

SELECT bldg_code AS "Building No", capacity Seat
FROM location
ORDER BY bldg_code;

# Column Alias

```
SELECT bldg_code AS "Building No", capacity Seat
FROM location
ORDER BY "Building No";
```

```
SELECT bldg_code AS Building No, capacity Seat
FROM location
ORDER BY Building No;
```

## Column Alias

```
SELECT bldg_code AS Building, capacity Seat
FROM location
ORDER BY "Building";

SELECT bldg_code AS "Building", capacity Seat
FROM location
ORDER BY Building;
```

## Column Alias

```
SELECT bldg_code AS Building, capacity Seat
FROM location
WHERE Building = 'BUS';
```

## Do it yourself

1. Display the last name and salary of employees who has the last name between A and L only, order in ascending order of last name.
2. Display the first name of all students in which last letter of the name is “a” or “l”.
3. Calculate the age of each student in the year 2022, rename the column as ‘Age’. Sort the result from the youngest to the oldest.

- Try the exercise given.