DQL

V P JAYACHITRA

Syntax

SELECT [DISTINCT | ALL] select_list

FROM table_reference

[WHERE search_condition]

[GROUP BY group_by_expression]

[HAVING search_condition]

[ORDER BY order expression [ASC | DESC]];

Logical Query Processing Order:

1. FROM: Identify source tables
2. WHERE: Apply row-level filters
3. GROUP BY: Group rows by criteria
4. HAVING: Filter grouped results
5. SELECT: Project columns/expressions
6. DISTINCT: Eliminate duplicates
7. ORDER BY: Sort final result set

Syntactic Writing Order

```
SELECT → FROM → WHERE → GROUP BY → HAVING → ORDER BY
```

ORDER BY Clause

- ORDER BY clause sorts query results based on one or more attributes in ascending or descending order.

```
SELECT column_list
FROM table_name
[WHERE condition]
ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...;
```

ORDER BY Clause

```
SELECT column_list
FROM table_name
[WHERE condition]
ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...;
```

Sort Order Modifiers:

- ASC: Ascending order (default)
- DESC: Descending order
- NULLS FIRST: NULL values appear first
- NULLS LAST: NULL values appear last (ASC default)

ORDER BY Clause

Query 1: Single column ascending sort (default)

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
ORDER BY Lname;
```

ORDER BY Clause

- Query 2: Descending order)

```
SELECT Fname, Lname, Salary

FROM EMPLOYEE

ORDER BY Salary DESC;
```

Query 3: Multi-column sort (hierarchical)

```
SELECT Fname, Lname, Salary, Dno
FROM EMPLOYEE
ORDER BY Dno ASC, Salary DESC;
```

First sort by Dno ascending, then within each department sort by Salary descending
Dno 1: James(55000)
Dno 4: Jennifer(43000), Alicia(25000), Ahmad(25000)
Dno 5: Franklin(40000), Ramesh(38000),

ORDER BY Clause

- Query 4: Sort by column position

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
ORDER BY 3 DESC;
```

Query 5: Sort by computed expression

```
SELECT Fname, Lname, Salary * 12 AS Annual_Salary
 FROM EMPLOYEE
 ORDER BY Salary * 12 DESC;
```

Query 6: NULL handling in Oracle

```
SELECT Fname, Lname, Super_ssn
FROM EMPLOYEE
ORDER BY Super_ssn NULLS LAST;
```

ORDER BY Clause

- ORDER BY ASC:
  - NULLs appear LAST
- ORDER BY DESC:
  - NULLs appear FIRST
- Important Properties:
  - ORDER BY executes LAST (after SELECT)
  - Can reference column aliases
  - Can sort by columns not in SELECT list
  - Multiple sort keys processed left-to-right

Aggregate Functions - Overview

- Aggregate functions compute single value from collection of tuples, implementing mathematical operations on multisets.

COUNT(*) - Count all tuples (with NULLs)

COUNT(column) - Count non-NULL values

SUM(column) - Sum of numeric values

AVG(column) - Arithmetic mean

MAX(column). - Maximum value

MIN(column) - Minimum value

Aggregate Functions - Overview

Characteristics:

- Input: Multiple tuples (multiset)
- Output: Single scalar value
- NULL values ignored (except COUNT(*))
- Used with GROUP BY for subgroup aggregation
- Cannot appear in WHERE clause

Aggregate Functions - Overview

Query 1: Count total employees

```
SELECT COUNT(*) AS Total_Employees
FROM EMPLOYEE;
```

Result: 8 (includes all tuples)

Query 2: Average salary calculation

```
SELECT AVG(Salary) AS Average_Salary
FROM EMPLOYEE;
```

Result:

(30000+40000+25000+43000+38000+25000+25000+55000)/8

= 35125

Query 3: Maximum and minimum values

```
SELECT MAX(Salary) AS Highest_Salary, MIN(Salary) AS
Lowest_Salary FROM EMPLOYEE;
```

Result: MAX = 55000 (James Borg), MIN = 25000

Aggregate Functions - Overview

Query 4: Sum aggregation

```
SELECT SUM(Salary) AS Total_Payroll
FROM EMPLOYEE;
```

Result: 281000 (sum of all salaries)

Query 5: Count non-NULL values

```
SELECT COUNT(Super_ssn) AS With_Supervisor
FROM EMPLOYEE;
```

Result: 7 (excludes James Borg's NULL supervisor)

Query 6: Count distinct values

```
SELECT COUNT(DISTINCT Dno) AS Num_Departments
FROM EMPLOYEE;
```

Result: 3 (departments 1, 4, 5)

Aggregate Functions - Overview

NULL Handling Rules:

- COUNT(*): Counts all tuples including NULL
- COUNT(column): Counts only non-NULL values
- SUM, AVG, MAX, MIN: Ignore NULL values
- If all values NULL: SUM returns NULL, COUNT returns 0

Restriction:

- Aggregate functions CANNOT be used in WHERE clause
- Use HAVING clause for aggregate-based filtering

GROUP BY Clause - Partitioning

- GROUP BY partitions relation into subsets based on grouping attributes, applying aggregate functions to each partition.

Syntax:

```
SELECT grouping_columns, aggregate_function(column)
FROM table_name
 [WHERE condition]
GROUP BY grouping_columns
[HAVING group_condition]
[ORDER BY columns];
```

GROUP BY Clause

- Query 1: Employee count per department

```
SELECT Dno, COUNT(*) AS Num_Employees
 FROM EMPLOYEE GROUP BY Dno;
```

Dno	Num_Employees
1	1
4	3
5	4

GROUP BY Clause

- Query 2: Average salary by department

```
SELECT Dno, AVG(Salary) AS Avg_Salary
FROM EMPLOYEE
 GROUP BY Dno;
```

Result:

Dno 1: 55000.00

Dno 4: 31000.00

Dno 5: 33250.00

GROUP BY Clause

- **Query 3: Payroll summary per department**

```
SELECT Dno, SUM(Salary) AS Total_Payroll
FROM EMPLOYEE
 GROUP BY Dno;
```

Result:

Dno 1: 55000

Dno 4: 93000

Dno 5: 133000

GROUP BY Clause

Query 4: Salary range by department

```
SELECT Dno, MIN(Salary) AS Min_Salary, MAX(Salary) AS Max_Salary, MAX(Salary) - MIN(Salary) AS Salary_Range
FROM EMPLOYEE
GROUP BY Dno;
```

Query 5: Gender distribution

```
SELECT gender, COUNT(*) AS Employee_Count
FROM EMPLOYEE
GROUP BY gender;
```

Result: M: 5, F: 3

GROUP BY Clause

Query 6: Multi-attribute grouping

```
SELECT Dno, gender, COUNT(*) AS Count
FROM EMPLOYEE
GROUP BY Dno, gender
 ORDER BY Dno, gender;
```

Result: (1,M,1), (4,F,1), (4,M,2), (5,F,1), (5,M,3)

GROUP BY Clause

Fundamental Rule:

Every column in SELECT must be either:

1. Listed in GROUP BY clause, OR
2. Inside an aggregate function

INVALID:

```
SELECT Dno, Fname, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;
```

(Fname not in GROUP BY and not aggregated)

HAVING Clause - Group Filtering

- HAVING clause filters groups created by GROUP BY based on aggregate conditions. WHERE filters tuples; HAVING filters groups.

Syntax:

```
SELECT grouping_columns, aggregate_function(column)
FROM table_name
 [WHERE tuple_condition]
GROUP BY grouping_columns
HAVING group_condition
[ORDER BY columns];
```

HAVING Clause - Group Filtering

- Query 1: Departments with more than 2 employees

```
SELECT Dno, COUNT(*) AS Num_Employees
FROM EMPLOYEE
GROUP BY Dno
 HAVING COUNT(*) > 2;
```

Result:

Dno	Num_Employees
4	3
5	4

HAVING Clause - Group Filtering

- Query 2: Departments with high average salary

```
SELECT Dno, AVG(Salary) AS Avg_Salary
 FROM EMPLOYEE
 GROUP BY Dno
  HAVING AVG(Salary) > 35000;
```

Result:

Dno	Avg_Salary
1	55000

HAVING Clause - Group Filtering

- Query 3: High-payroll departments

```
SELECT Dno, SUM(Salary) AS Total_Payroll
FROM EMPLOYEE
GROUP BY Dno
HAVING SUM(Salary) > 100000;
```

Result:

Dno	Total_Payroll
5	133000

HAVING Clause - Group Filtering

- **Query 4: Combined WHERE and HAVING**

```
SELECT Dno, AVG(Salary) AS Avg_Male_Salary
FROM EMPLOYEE
WHERE gender = 'M’
GROUP BY Dno
HAVING AVG(Salary) > 30000;
```

HAVING Clause - Group Filtering

- **Query 5: Complex HAVING conditions**

```
SELECT Dno, COUNT(*) AS Num_Employees, AVG(Salary) AS Avg_Salary
FROM EMPLOYEE
GROUP BY Dno
 HAVING COUNT(*) >= 2 AND AVG(Salary) > 30000;
```

Efficiency Consideration:

WHERE is more efficient than HAVING for non-aggregate conditions

because it reduces tuples before grouping.

Better: WHERE Salary > 25000 (filters early)

Worse: HAVING Salary > 25000 (processes all then filters)

Retrieve all employees in department 5

A) **SELECT Fname, Lname, Dno**
FROM EMPLOYEE
HAVING Dno = 5;

B) SELECT Fname, Lname, Dno
FROM EMPLOYEE
WHERE Dno = 5;

C) SELECT Fname, Lname
FROM EMPLOYEE
WHERE Dno = 5
GROUP BY Dno;

D) SELECT DISTINCT Fname, Lname, Dno
FROM EMPLOYEE
ORDER BY Dno = 5;

Retrieve all employees in department 5

A) SELECT Fname, Lname, Dno
FROM EMPLOYEE
HAVING Dno = 5;

B) SELECT Fname, Lname, Dno
FROM EMPLOYEE
WHERE Dno = 5;

C) SELECT Fname, Lname
FROM EMPLOYEE
WHERE Dno = 5
GROUP BY Dno;

D) SELECT DISTINCT Fname, Lname, Dno
FROM EMPLOYEE
ORDER BY Dno = 5;

ANSWER: B

Find female employees earning over 30,000

A)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F' OR Salary > 30000;
```

B)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F' AND Salary > 30000;
```

C)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F'
  HAVING Salary > 30000;
```

D)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F' AND Salary >= 30000;
```

Find female employees earning over 30,000

A)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F' OR Salary > 30000;
```

B)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F' AND Salary > 30000;
```

C)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F'
  HAVING Salary > 30000;
```

D)
```
SELECT Fname, Lname, Sex, Salary
  FROM EMPLOYEE
  WHERE Sex = 'F' AND Salary >= 30000;
```

ANSWER: B

Employees with salary between 30K and 40K (inclusive)

A)
```
SELECT Fname, Lname, Salary
  FROM EMPLOYEE
  WHERE Salary > 30000 AND Salary < 40000;
```

B)
```
SELECT Fname, Lname, Salary
  FROM EMPLOYEE
  WHERE Salary >= 30000 AND Salary <= 40000;
```

C)
```
SELECT Fname, Lname, Salary
  FROM EMPLOYEE
  WHERE Salary BETWEEN 30000 AND 40000;
```

D) Both B and C are correct

Employees with salary between 30K and 40K (inclusive)

A) SELECT Fname, Lname, Salary

FROM EMPLOYEE

WHERE Salary > 30000 AND Salary < 40000;

B) SELECT Fname, Lname, Salary

FROM EMPLOYEE

WHERE Salary >= 30000 AND Salary <= 40000;

C) SELECT Fname, Lname, Salary

FROM EMPLOYEE

WHERE Salary BETWEEN 30000 AND 40000;

D) Both B and C are correct

ANSWER: D

Find employees whose last name starts with 'W' or 'S'

A)
```
SELECT Fname, Lname
  FROM EMPLOYEE
  WHERE Lname LIKE 'W%' OR Lname LIKE 'S%';
```

B)
```
SELECT Fname, Lname
  FROM EMPLOYEE
  WHERE Lname LIKE 'W%' OR 'S%';
```

C)
```
SELECT Fname, Lname
  FROM EMPLOYEE
  WHERE Lname IN ('W%', 'S%');
```

D) Both B and C are correct

Find employees whose last name starts with 'W' or 'S'

A) SELECT Fname, Lname
FROM EMPLOYEE
WHERE Lname LIKE 'W%' OR Lname LIKE 'S%';

B) SELECT Fname, Lname
FROM EMPLOYEE
WHERE Lname LIKE 'W%' OR 'S%';

C) SELECT Fname, Lname
FROM EMPLOYEE
WHERE Lname IN ('W%', 'S%');

D) Both B and C are correct

ANSWER: A

Find employees who have a supervisor

A)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE Super_ssn != NULL;
```

B)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE Super_ssn <> NULL;
```

C)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE Super_ssn IS NOT NULL;
```

D)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE NOT Super_ssn = NULL;
```

Find employees who have a supervisor

A)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE Super_ssn != NULL;
```

B)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE Super_ssn <> NULL;
```

C)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE Super_ssn IS NOT NULL;
```

D)
```
SELECT Fname, Lname, Super_ssn
  FROM EMPLOYEE
  WHERE NOT Super_ssn = NULL;
```

ANSWER: C

Count employees in each department

A) **SELECT Dno, COUNT(*)**
FROM EMPLOYEE;

B) SELECT Dno, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

C) SELECT Dno, COUNT(Fname)
FROM EMPLOYEE
WHERE Dno IS NOT NULL;

D) SELECT DISTINCT Dno, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

Count employees in each department

A) SELECT Dno, COUNT(*)
FROM EMPLOYEE;

B) SELECT Dno, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

C) SELECT Dno, COUNT(Fname)
FROM EMPLOYEE
WHERE Dno IS NOT NULL;

D) SELECT DISTINCT Dno, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

ANSWER: B

Valid examples:

```
SELECT Dno, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;
```

Dno in GROUP BY

```
SELECT COUNT(*)
FROM EMPLOYEE;
```

No non-aggregate columns

```
SELECT Dno, Sex, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno, Sex;
```

Both Dno and Sex in GROUP BY

Invalid examples:

```
SELECT Dno, Fname, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;
```

Fname NOT in GROUP BY!

```
SELECT Dno, COUNT(*)
FROM EMPLOYEE;
```

Missing GROUP BY!

Calculate average salary for each department

A) SELECT Dno, SUM(Salary) / COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

B) SELECT Dno, AVG(Salary) AS Avg_Salary
FROM EMPLOYEE
GROUP BY Dno;

C) SELECT AVG(Salary) AS Avg_Salary
FROM EMPLOYEE
GROUP BY Dno;

D) SELECT Dno, Salary / COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

Calculate average salary for each department

A) SELECT Dno, SUM(Salary) / COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

B) SELECT Dno, AVG(Salary) AS Avg_Salary
FROM EMPLOYEE
GROUP BY Dno;

C) SELECT AVG(Salary) AS Avg_Salary
FROM EMPLOYEE
GROUP BY Dno;

D) SELECT Dno, Salary / COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;

CORRECT ANSWER: B

comments

What if all values in group are NULL?

AVG() returns NULL

Difference between AVG(column) and SUM(column)/COUNT(*)?

- AVG(column) excludes NULLs from count
- SUM/COUNT(*) includes NULLs in COUNT(*)
- AVG() excludes NULL values

Find departments with more than 2 employees

Expected output

Dno	Emp_Count
4	3
5	4

Find departments with more than 2 employees

A)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
WHERE COUNT(*) > 2
GROUP BY Dno;
```

B)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
GROUP BY Dno
HAVING COUNT(*) > 2;
```

C)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
GROUP BY Dno
WHERE COUNT(*) > 2;
```

D)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
GROUP BY Dno
HAVING Emp_Count > 2;
```

Find departments with more than 2 employees

A)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
WHERE COUNT(*) > 2
GROUP BY Dno;
```

B)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
GROUP BY Dno
HAVING COUNT(*) > 2;
```

C)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
GROUP BY Dno
WHERE COUNT(*) > 2;
```

D)
```
SELECT Dno, COUNT(*) AS Emp_Count
FROM EMPLOYEE
GROUP BY Dno
HAVING Emp_Count > 2;
```

ANSWER: B

- Aliases available in:

  ORDER BY (executes after SELECT)

- Aliases not available in:

  HAVING ✗ (executes before SELECT)

  WHERE ✗ (executes before SELECT)

  GROUP BY ✗ (executes before SELECT)

Count employees by department AND gender

Expected output

Dno	Gender	Count
1	M	1
4	F	1
4	M	2
5	F	1
5	M	3

Count employees by department AND gender

A)
```
SELECT Dno, Gender, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;
```

B)
```
SELECT Dno, Gender, COUNT(*)
FROM EMPLOYEE
GROUP BY Gender;
```

C)
```
SELECT Dno, Gender, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno, Gender;
```

D)
```
SELECT Dno, Gender, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno, Gender;
```

Count employees by department AND gender

A)
```
SELECT Dno, Gender, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno;
```

B)
```
SELECT Dno, Gender, COUNT(*)
FROM EMPLOYEE
GROUP BY Gender;
```

C)
```
SELECT Dno, Gender, COUNT(*)
FROM EMPLOYEE
GROUP BY Dno, Gender;
```

D)
```
SELECT Dno, COUNT(*), Gender
FROM EMPLOYEE
GROUP BY Dno, Gender;
```

ANSWER: C (and D)

Departments with 2+ employees AND avg salary > 30000

Expected output

- Dno Count Avg_Salary
- 5 4 33250.00

Departments with 2+ employees AND avg salary > 30000

A)
```
SELECT Dno, COUNT(*), AVG(Salary)
FROM EMPLOYEE
WHERE COUNT(*) >= 2 AND AVG(Salary) > 30000
GROUP BY Dno;
```

B)
```
SELECT Dno, COUNT(*), AVG(Salary)
FROM EMPLOYEE
GROUP BY Dno
HAVING COUNT(*) >= 2 AND AVG(Salary) > 30000;
```

C)
```
SELECT Dno, COUNT(*), AVG(Salary)
FROM EMPLOYEE
GROUP BY Dno
HAVING COUNT(*) >= 2 OR AVG(Salary) > 30000;
```

D)
```
SELECT Dno, COUNT(*) AS Cnt, AVG(Salary) AS Avg_Sal
FROM EMPLOYEE
GROUP BY Dno
HAVING Cnt >= 2 AND
```

Departments with 2+ employees AND avg salary > 30000

A)
```
SELECT Dno, COUNT(*), AVG(Salary)
  FROM EMPLOYEE
  WHERE COUNT(*) >= 2 AND AVG(Salary) > 30000
  GROUP BY Dno;
```

B)
```
SELECT Dno, COUNT(*), AVG(Salary)
  FROM EMPLOYEE
  GROUP BY Dno
  HAVING COUNT(*) >= 2 AND AVG(Salary) > 30000;
```

C)
```
SELECT Dno, COUNT(*), AVG(Salary)
  FROM EMPLOYEE
  GROUP BY Dno
  HAVING COUNT(*) >= 2 OR AVG(Salary) > 30000;
```

D)
```
SELECT Dno, COUNT(*) AS Cnt, AVG(Salary) AS Avg_Sal
  FROM EMPLOYEE
  GROUP BY Dno
  HAVING Cnt >= 2 AND
```

ANSWER: B

Find department with SECOND highest average salary

Expected Output:

Dno	Avg_Salary
5	33250.00

Subquery

- A subquery is a complete SELECT statement that is nested inside another SELECT, INSERT, UPDATE, or DELETE statement, or inside another subquery
- A subquery is a query within another query

```
SELECT column(s)
FROM table
WHERE column OPERATOR (
   SELECT column
   FROM table
   [WHERE condition]
);
```

Subquery

- Rule 1: Subquery Must Be Enclosed in Parentheses
- Correct:WHERE Salary > (SELECT AVG(Salary) FROM EMPLOYEE)
- Wrong: WHERE Salary > SELECT AVG(Salary) FROM EMPLOYEE -- Missing ( )

Subquery

- Rule 2: Subquery Must Return Compatible Data Type
- Correct: WHERE Salary > (SELECT AVG(Salary) FROM EMPLOYEE)

  Returns NUMBER
- Wrong: WHERE Salary > (SELECT Fname FROM EMPLOYEE)

  Returns STRING!

Subquery

- Rule 3: Subquery for = Must Return Single Value
- Correct: WHERE Dno = (SELECT Dnumber FROM DEPARTMENT WHERE Dname='Research')

  returns 1 value:5
- Wrong: WHERE Dno = (SELECT Dnumber FROM DEPARTMENT) Returns 3 values

Subquery

- ORDER BY Not Allowed in Subquery (Usually)

```
WHERE Salary > (
    SELECT AVG(Salary)
    FROM EMPLOYEE
    ORDER BY Salary  -- Meaningless for single value!
)
```

Subquery

Who scored higher than the class average?

- Step 1: "What is the class average?"
- 75%
- Step 2: "Who scored higher than 75?“

In SQL:

- Step 1 = SUBQUERY
- Step 2 = OUTER QUERY

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
WHERE Salary > AVG(Salary);  -- Will this work?
```

Subquery

Who scored higher than the class average?

- Step 1: "What is the class average?"
- 75%
- Step 2: "Who scored higher than 75?“

In SQL:

- Step 1 = SUBQUERY
- Step 2 = OUTER QUERY

```
SELECT Fname, Lname, Salary
FROM EMPLOYEE
WHERE Salary > (SELECT AVG(Salary) FROM
EMPLOYEE);
```

Subquery

Subquery : Query nested inside another query

Inner Query : The nested query (same as subquery)

Outer Query : The main query containing subquery

Nested Query : Another name for subquery

When Do We Need Subqueries?

- Comparing to Aggregate Values
- Checking Existence
- Finding Top-N Values
- Dynamic Filtering

When Do We Need Subqueries?

- Write a query to find products priced above the average price.

```
SELECT ProductID, Name, Price
FROM PRODUCTS
WHERE Price > (SELECT AVG(Price) FROM PRODUCTS);
```

Syntax Patterns by Operator

- Single Value Comparison

```
WHERE column {=|>|<|>=|<=|<>} (
    SELECT AGG_FUNCTION(column)
    FROM table
)
```

```
WHERE Salary > (SELECT AVG(Salary) FROM EMPLOYEE)

WHERE Price = (SELECT MAX(Price) FROM PRODUCTS)

WHERE Age < (SELECT MIN(Age) FROM STUDENTS)
```

Syntax Patterns by Operator

- IN Operator (Multiple Values)

```
WHERE column IN (
    SELECT column
    FROM table
    [WHERE condition]
)
```

```
WHERE Dno IN (SELECT Dnumber FROM DEPARTMENT WHERE
Location='Houston')
WHERE StudentID IN (SELECT StudentID FROM ENROLLMENTS WHERE
Grade='A')
```

Syntax Patterns by Operator

- ANY/ALL Operators

WHERE column OPERATOR {ANY|ALL} (

    SELECT column

    FROM table

)

```
WHERE Salary > ANY (SELECT Salary FROM EMPLOYEE WHERE Dno=5)
WHERE Price <= ALL (SELECT Price FROM PRODUCTS WHERE CategoryID=2)
```

Syntax Patterns by Operator

- EXISTS Operator

```
WHERE EXISTS (
    SELECT *
    FROM table
    WHERE condition
)
```

WHERE EXISTS (SELECT * FROM DEPENDENT WHERE Essn=E.Ssn)