

SQL Queries using **SELECT**



```
A query in SQL has the form:
```

```
SELECT (list of columns or expressions)
```

```
FROM (list of tables)
```

WHERE (filter conditions)

GROUP BY (columns)

ORDER BY (columns)

Notes:

- 1) Separate the list of columns/expressions and list of tables by commas.
- 2) The "*" is used to select all columns.
- 3) Only SELECT required. FROM, WHERE, GROUP BY, ORDER BY are optional.





The SELECT statement can be mapped directly to relational algebra.

SELECT
$$A_1$$
, A_2 , ..., A_n
FROM R_1 , R_2 , ..., R_m
WHERE P

is equivalent to:

$$\Pi_{A_1,A_2,...,A_n}(\sigma_P(R_1 \times R_2 \times ... \times R_m))$$





emp Table

<u>eno</u>	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

proj Table

<u>pno</u>	pname	budget	dno
P1	Instruments	150000	D1
P2	DB Develop	135000	D2
P3	Budget	250000	D3
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

workson Table

<u>eno</u>	<u>pno</u>	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

dept Table

<u>dno</u>	dname	mgreno
D1	Management	E8
D2	Consulting	E7
D3	Accounting	E5
D4	Development	null



SQL: Retrieving Only Some of the Columns

The *projection operation* creates a new table that has some of the columns of the input table. In SQL, provide the table in the FROM clause and the fields in the output in the SELECT.

Example: Return only the eno field from the emp table:

SELECT eno

FROM emp

emp Table

<u>eno</u>	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

Result

eno	
E1	
E2	
E3	
E4	
E5	
E6	
E7	
E8	

SQL Projection Examples



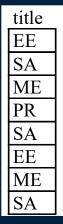
emp Table

<u>eno</u>	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

SELECT eno,ename
FROM emp

<u>eno</u>	ename
E1	J. Doe
E2	M. Smith
E3	A. Lee
E4	J. Miller
E5	B. Casey
E6	L. Chu
E7	R. Davis
E8	J. Jones

SELECT title
FROM emp



- Notes: 1) Duplicates are not removed during SQL projection.
 - 2) SELECT * will return all columns.





Question: Given this table and the query:

```
SELECT eno, ename, salary
FROM emp
```

How many columns are returned?

- **A)** 0
- B) 1
- **C)** 2
- **D)** 3
- E) 4

emp Table

<u>eno</u>	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000





Question: Given this table and the query:

SELECT salary
FROM emp

How many rows are returned?

A) 0

B) 2

C) 4

D) 8

emp Table

<u>eno</u>	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000





One major difference between SQL and relational algebra is that relations in SQL are bags instead of sets.

• It is possible to have two or more identical rows in a relation.

Consider the query: Return all titles of employees.

SELECT title
FROM emp

emp Table

<u>eno</u>	ename	bdate	title	salary	supereno	dno
E1	J. Doe	01-05-75	EE	30000	E2	null
E2	M. Smith	06-04-66	SA	50000	E5	D3
E3	A. Lee	07-05-66	ME	40000	E7	D2
E4	J. Miller	09-01-50	PR	20000	E6	D3
E5	B. Casey	12-25-71	SA	50000	E8	D3
E6	L. Chu	11-30-65	EE	30000	E7	D2
E7	R. Davis	09-08-77	ME	40000	E8	D1
E8	J. Jones	10-11-72	SA	50000	null	D1

Result

EE
SA
ME
PR
SA
EE
ME
SA
SA

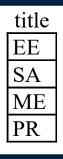




To remove duplicates, use **DISTINCT** clause in the SQL statement:

```
SELECT DISTINCT title
FROM emp
```

Result







Question: Given this table and the query:

SELECT DISTINCT a, b

FROM R

How many rows are returned?

A) 1

B) 3

C) 4

D) 6

R Table

a	b	С			
1	1	А			
1	2	В			
1	1	А			
3	1	С			
2	2	А			
2	2	В			





Question: Using the proj table, write these three queries:

- 1) Show all rows and all columns.
- 2) Show all rows but only the pno column.
- 3) Show all rows but only the pno and budget columns.
- 4) Show unique budget values.





The *selection operation* creates a new table with some of the rows of the input table. A condition specifies which rows are in the new table. The condition is similar to an if statement.

Example: Return the projects in department 'D2':

SELECT pno, pname, budget, dno

FROM proj

WHERE dno = 'D2'

proj Table

<u>pno</u>	pname	budget	dno
P1	Instruments	150000	D1
P2	DB Develop	135000	D2
Р3	Budget	250000	D3
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

Result

pno	pname	budget	dno
P2	DB Develop	135000	D2
P4	Maintenance	310000	D2
P5	CAD/CAM	500000	D2

Algorithm: Scan each tuple and check if matches condition in WHERE clause.





The condition in a selection statement specifies which rows are included. It has the general form of an if statement.

The condition may consist of attributes, constants, comparison operators (<, >, =, !=, <=, >=), and logical operators (AND, OR, NOT).





emp Table

<u>eno</u>	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

SELECT *

FROM emp

WHERE title = 'EE'

eno	ename	title	salary
E1	J. Doe	EE	30000
E6	L. Chu	EE	30000

eno	ename	title	salary
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000





Question: Given this table and the query:

```
SELECT *
FROM emp
```

WHERE title='SA'

How many rows are returned?

- **A)** 0
- B) 1
- **C)** 2

D) 3

emp Relation

<u>eno</u>	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000





Question: Given this table and the query:

```
SELECT *
FROM emp
WHERE salary > 50000 or title='PR'
```

emp Table

How many rows are returned?	<u>eno</u>	ename	title	salary
riow many rows are retained.	eno E1	J. Doe	EE	30000
A) 0	E2	M. Smith	SA	50000
	E3	A. Lee	ME	40000
B) 1	E4	J. Miller	PR	20000
C) 2	E5	B. Casey	SA	50000
C) 2	E6	L. Chu	EE	30000
D) 3	E7	R. Davis	ME	40000
	E8	J. Jones	SA	50000





Question: Write these queries:

- 1) Return all projects with budget > \$250000.
- 2) Show the pno and pname for projects in dno = 'D1'.
- 3) Show pno and dno for projects in dno='D1' or dno='D2'.
- 4) Return the employee numbers who make less than \$30000.
- 5) Return list of workson responsibilities (resp) with no duplicates.
- 6) Return the employee (names) born after July 1, 1970 that have a salary > 35000 and have a title of 'SA' or 'PR'.





A join combines two tables by matching columns in each table.

workson Table

eno	<u>pno</u>	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36

proj Table

<u>pno</u>	pname	budget
P1	Instruments	150000
P2	DB Develop	135000
P3	CAD/CAM	250000
P4	Maintenance	310000
P5	CAD/CAM	500000

SELECT * FROM workson JOIN proj ON workson.pno = proj.pno

eno	pno	resp	hours	proj.pno	pname	budget
E1	P1	Manager	12	P1	Instruments	150000
E2	P1	Analyst	24	P1	Instruments	150000
E2	P2	Analyst	6	P2	DB Develop	135000
E3	P3	Consultant	10	P3	DB Develop	135000
E3	P4	Engineer	48	P4	Maintenance	310000
E4	P2	Programmer	18	P2	DB Develop	135000
E5	P2	Manager	24	P2	DB Develop	135000
E6	P4	Manager	48	P4	Maintenance	310000
E7	P3	Engineer	36	P3	CAD/CAM	250000





Listing multiple tables in the FROM clause separated by commas creates a cross product of tables. Must specify JOIN and ON or provide join condition in WHERE clause.

Goal: For each employee, return their name and department name.

Wrong! Cross Product

SELECT ename, dname

FROM emp, dept

Correct! JOIN-ON Clause

SELECT ename, dname

FROM emp JOIN dept

ON emp.dno = dept.dno

Correct! Join in WHERE

SELECT ename, dname

FROM emp, dept

WHERE emp.dno = dept.dno

Correct! Order does not matter.

SELECT ename, dname

FROM dept JOIN emp

ON emp.dno = dept.dno

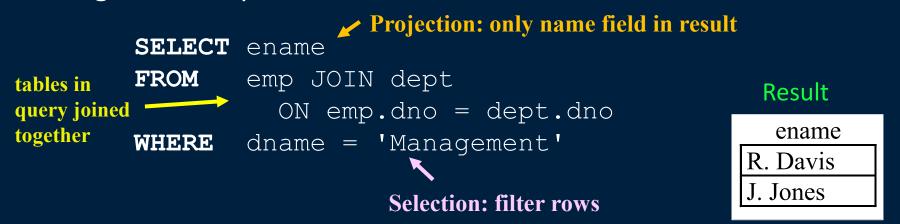




You can use join, selection, and projection in the same query.

• Recall: Projection returns columns listed in SELECT, selection filters out rows using condition in WHERE, and join combines tables in FROM using a condition.

Example: Return the employee names who are assigned to the 'Management' department.





Three Table Join Query Example

Return all projects who have an employee working on them whose title is 'EE':

```
SELECT pname
FROM emp JOIN workson ON emp.eno = workson.eno
           JOIN proj ON workson.pno = proj.pno
      emp.title = 'EE'
WHERE
SELECT
      pname
FROM
       emp, proj, workson
       emp.title = 'EE' and workson.eno = emp.eno
WHERE
        and workson.pno = proj.pno
```

Note: Parentheses () can be used to specify order of joins when using JOIN-ON.





Question: What query would return the name and salary of employees working on project 'P3':

```
A) SELECT ename, salary
FROM emp, workson
WHERE emp.eno = workson.eno and pno = 'P3'
```

B) SELECT ename, salary
FROM emp, workson, proj
WHERE emp.eno = workson.eno and pno = "P3"





The query result returned is not ordered on any column by default. We can order the data using the **ORDER BY** clause:

```
SELECT ename, salary, bdate
FROM emp
WHERE salary > 30000
ORDER BY salary DESC, ename ASC;
```

- 'ASC' sorts the data in ascending order, and 'DESC' sorts it in descending order. The default is 'ASC'.
- The order of sorted attributes is significant. The first column specified is sorted on first, then the second column is used to break any ties, etc.





If you only want the first N rows, use a LIMIT clause:

```
SELECT ename, salary FROM emp
ORDER BY salary DESC LIMIT 5
```

To start from a row besides the first, use OFFSET:

```
SELECT eno, salary FROM emp
ORDER BY eno DESC
LIMIT 3 OFFSET 2
```

- LIMIT improves performance by reducing amount of data processed and sent by the database system.
- OFFSET 0 is first row, so OFFSET 2 would return the 3rd row.
- LIMIT/OFFSET syntax support differs between databases.





Question: Write these queries:

- 1) Return all projects with budget < \$500000 sorted by budget descending.
- 2) List only the top 5 employees by salary descending. Show only their name and salary.
- 3) List each project pno, dno, pname, and dname ordered by dno ascending then pno ascending. Only show projects if department name > 'D'. Note: This query will require a join.
- 4) Return the list of project names for the department with name 'Consulting'.
- 5) Return workson records (eno, pno, resp, hours) where project budget is > \$50000 and hours worked is < 20.
- 6) **Challenge:** Return a list of all department names, the names of the projects of that department, and the name of the manager of each department.

Calculated Fields



Expressions are allowed in SELECT clause to perform calculations.

• When an expression is used to define an attribute, the DBMS gives the attribute a unique name such as coll, col2, etc.

Example: Return how much employee 'A. Lee' will get paid for his work on each project.

Result

ename	pname	col3
A. Lee	Budget	192.31
A. Lee	Maintenance	923.08





Often it is useful to rename an attribute in the final result (especially when using calculated fields). Renaming is accomplished using the keyword AS:

Result

ename	pname	pay
A. Lee	Budget	192.31
A. Lee	Maintenance	923.08

Note: AS keyword is optional.



Renaming and Aliasing Tables

Renaming is also used when two or more copies of the same table are in a query. Using *aliases* allows you to uniquely identify what table you are talking about.

Example: Return the employees and their managers where the managers make less than the employee.

```
SELECT E.ename, M.ename
FROM emp as E JOIN emp as M ON E.supereno = M.eno
WHERE E.salary > M.salary
```





Used when the condition in the WHERE clause will request tuples where one attribute value must be in a *range* of values.

Example: Return the employees who make at least \$20,000 and less than or equal to \$45,000.

```
FROM emp
WHERE salary >= 20000 and salary <= 45000</pre>
```

We can use the keyword **BETWEEN** instead:

```
SELECT ename
FROM emp
WHERE salary BETWEEN 20000 and 45000
```





For strings, the **LIKE** operator is used to search for partial matches.

 Partial string matches are specified by using either "%" that replaces zero or more characters or underscore " " that replaces a single character.

Example: Return all employee names that start with 'A'.

```
SELECT ename
FROM emp
WHERE ename LIKE 'A%'
```

Example: Return all employee names who have a first name that starts with 'J' and whose last name is 3 characters long.

```
FROM emp
WHERE ename LIKE 'J. _ _ _ '
```





Warning: Do not use the LIKE operator if you do not have to.

It is often an inefficient operation as the DBMS may not be able to optimize lookup using LIKE as it can for equal (=) comparisons. The result is the DBMS often has to examine ALL TUPLES in the relation.

In almost all cases, adding indexes will **not** increase the performance of LIKE queries because the indexes cannot be used.

• Most indexes are implemented using B-trees that allow for fast equality searching and efficient range searches.





To specify that an attribute value should be in a given set of values, the IN keyword is used.

• Example: Return employees who are in one of the departments {'D1', 'D2', 'D3'}.

```
SELECT ename
FROM emp
WHERE dno IN ('D1','D2','D3')
```

Note that this is equivalent to using OR:

```
SELECT ename
FROM emp
WHERE dno = 'D1' OR dno = 'D2' OR dno = 'D3'
```

We will see more uses of IN and NOT IN with nested subqueries.





Remember NULL indicates that an attribute does not have a value. To determine if an attribute is NULL, we use the clause **IS NULL**.

• Note that you should not test NULL values using = and <>.

Example: Return all employees who are not in a department.

```
SELECT ename
FROM emp
WHERE dno IS NULL
```

Example: Return all departments that have a manager.

```
FROM dept
WHERE mgreno IS NOT NULL
```

Set Operations



Union, intersection, and difference combine results of two queries.

- UNION, INTERSECT, EXCEPT, UNION ALL (returns all rows)
- *Union-compatible*: same # of attributes and compatible data types. Do not need to have the same name.

Example: Return the employees who are either directly supervised by 'R. Davis' or directly supervised by 'M. Smith'.

```
(SELECT E.ename
FROM emp as E JOIN emp as M ON E.supereno = M.eno
WHERE M.ename='R. Davis')
UNION
(SELECT E.ename
FROM emp as E JOIN emp as M ON E.supereno = M.eno
WHERE M.ename='M. Smith')
```

Set Operations Examples

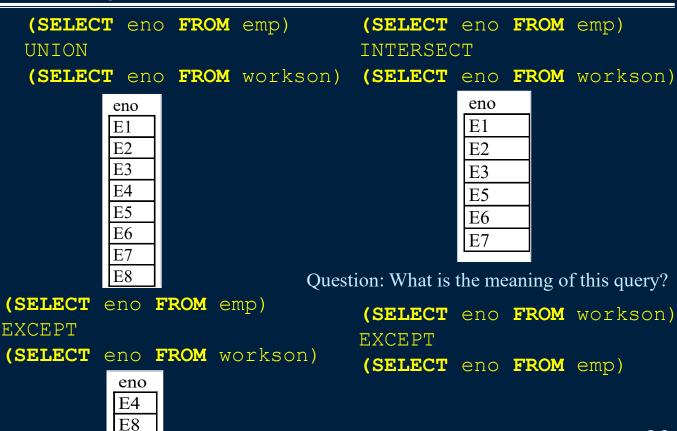


emp Table

<u>eno</u>	ename	title	salary
E1	J. Doe	EE	30000
E2	M. Smith	SA	50000
E3	A. Lee	ME	40000
E4	J. Miller	PR	20000
E5	B. Casey	SA	50000
E6	L. Chu	EE	30000
E7	R. Davis	ME	40000
E8	J. Jones	SA	50000

workson Table

<u>eno</u>	<u>pno</u>	resp	hours
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36





Set Operations Union-compatible Question

Question: Two tables have the same number of fields in the same order with the same types, but the names of some fields are different. True or false: The two tables are union-compatible.

A) true

B) false





The result of a select statement can be stored in a temporary table using the **INTO** keyword.

SELECT E.ename
INTO davisMgr
FROM emp as E JOIN emp as M ON E.supereno = M.eno
WHERE M.ename = 'R. Davis'





Question: What query would return the department names that do not have a manager or contain 'ent'.

```
A) SELECT dname
FROM dept
WHERE mgreno = NULL OR dname LIKE '_ent'
```

```
B) SELECT dname
FROM dept
WHERE mgreno IS NULL OR dname LIKE '%ent%'
```

Try it: SQL SELECT Expressions, LIKE, IS NULL

Question: Write these queries:

- 1) Calculate the monthly salary for each employee.
- 2) List all employee names who do not have a supervisor.
- 3) List all employee names where the employee's name contains an 'S' and workson responsibility that ends in 'ER'.
- 4) Return the list of employees (names) who make less than their managers and how much less they make.
- 5) Return only the top 3 project budgets in descending order.





Question: Write these queries:

- 1) Return the list of employees sorted by salary (desc) and then title (asc).
- 2) Return the employees (names) who either manage a department or manage another employee.
- 3) Return the employees (names) who manage an employee but do not manage a department.
- 4) Give a list of all employees who work on a project for the 'Management' department ordered by project number (asc).
- 5) Challenge: Return the projects (names) that have their department manager working on them.

Conclusion



The **SELECT** statement is used to query data and combines the operations of selection, projection, and join.

SELECT features covered:

- **SELECT** clause to provide column list and calculate expressions
- DISTINCT clause to eliminate duplicates
- FROM clause to list tables
- JOIN ON syntax to join tables on a join condition
- UNION, EXCEPT, INTERSECT set operations
- IS NULL for checking if column value is null
- ORDER BY clause for sorting output
- LIMIT/OFFSET for only reducing a part of the result set

Objectives



Translate English questions into SQL queries that may require:

- SELECT-FROM-WHERE syntax for selection, projection, and join
- renaming and aliasing including queries with multiple copies of the same relation
- ORDER BY
- LIMIT/OFFSET
- DISTINCT to eliminate duplicates
- UNION, EXCEPT, INTERSECT set operations
- IS NULL or IS NOT NULL
- LIKE string pattern matching

Read SQL queries to determine their output and English meaning

