Restricting and Sorting Data

The WHERE Clause & Comparison Operators

To limit the rows that are selected, we use WHERE and it always come after the FROM clause

```
SELECT *
FROM EMPLOYEES
WHERE DEPARTMENT_ID=90;
```

Character strings and date values are enclosed with single quotation marks.
Character values are case-sensitive and date values are format-sensitive.
The default date display format is DD-MON-RR.

Using WHERE in char column

```
SELECT *
FROM EMPLOYEES
WHERE FIRST_NAME='Steven'; -- the data is Case sensitive
```

Using WHERE in date column

```
SELECT *
FROM EMPLOYEES
WHERE HIRE_DATE = '17-OCT-03';
```

Using the comparison operators

```
SELECT *
FROM EMPLOYEES
WHERE HIRE_DATE > '17-OCT-03';
```

```
SELECT *
FROM EMPLOYEES
WHERE FIRST_NAME > 'Alberto'
ORDER BY FIRST_NAME;
```

Using Between and / IN / Like Operators

Using between and

```
SELECT * FROM EMPLOYEES
```

```
WHERE SALARY BETWEEN 10000 AND 20000; -- always the lower limit first, then higher limit
```

You can also use operators in varchar columns

```
SELECT * FROM EMPLOYEES
WHERE FIRST_NAME BETWEEN 'A' AND 'C'
ORDER BY FIRST_NAME;
```

Using IN operator

```
SELECT * FROM EMPLOYEES
WHERE SALARY In (10000, 25000, 17000); -- the order is not imporatant
```

Using LIKE operator and it usually comes with _ and %

- % means zero or more characters
- _ means one character

```
SELECT * FROM EMPLOYEES
WHERE FIRST_NAME LIKE 'S%'; -- All the first name which start with s
```

```
SELECT * FROM EMPLOYEES
WHERE FIRST_NAME LIKE '%s'; -- All the first name which end with s
```

```
SELECT * FROM EMPLOYEES
WHERE FIRST_NAME LIKE '%am%'; -- All the first name which include with am
```

```
SELECT * FROM EMPLOYEES
WHERE FIRST_NAME LIKE '_d%'; -- the first_name which has d in second letter
```

```
SELECT * FROM EMPLOYEES
WHERE FIRST_NAME LIKE '__s%'; -- the first_name which has s in third letter
```

Now suppose there is a value in any column contain _ or % (example job_id)

Example:

I need all the job_id which contain the string SA_

Let us add new job called SAP cons

If you try this select statement, it will pick all the job_id contain SA followed by any character

```
SELECT JOB_ID
FROM JOBS
WHERE JOB_ID LIKE 'SA_%';
```

JOB_ID
SAP cons
SA MAN
SA REP

This is the correct select statement using ESCAPE

```
SELECT JOB_ID
FROM JOBS
WHERE JOB_ID LIKE 'SA/_%' ESCAPE '/';
```

JOB_ID
SA MAN
SA REP

Using IS NULL / NOT / Not equal Operators

Pick all the employees who don't have commissions

```
SELECT *
FROM EMPLOYEES
WHERE COMMISSION_PCT IS NULL; -- don't use COMMISSION_PCT=' ' because this is not correct
```

You can also use NOT LIKE, NOT IN, IS NOT NULL, NOT BETWEEEN AND

```
SELECT *
FROM EMPLOYEES
WHERE EMPLOYEE_ID NOT IN (100,101);
```

```
SELECT *
FROM EMPLOYEES
WHERE COMMISSION_PCT IS NOT NULL;
```

```
SELECT *
FROM EMPLOYEES
WHERE FIRST_NAME NOT LIKE 'S%'; -- All the first name which does not start with S
```

The next 2 queries give the same result

```
SELECT *
FROM EMPLOYEES
WHERE DEPARTMENT_ID<>50;
```

```
SELECT *
FROM EMPLOYEES
WHERE DEPARTMENT_ID!=50;
```

Logical Operators (AND / OR / NOT)

Defining conditions using the logical operators:

AND requires both the component conditions to be true

```
SELECT employee_id, last_name, job_id, salary, DEPARTMENT_ID
FROM EMPLOYEES
WHERE SALARY > 10000
AND DEPARTMENT_ID=90;
```

OR requires either component condition to be true

```
SELECT employee_id, last_name, job_id, salary, DEPARTMENT_ID
FROM EMPLOYEES
WHERE SALARY > 10000
AND DEPARTMENT_ID=90;
```

Let's see this 3 AND statements

```
SELECT employee_id, last_name, job_id, salary, DEPARTMENT_ID, COMMISSION_PCT
FROM EMPLOYEES
WHERE SALARY >2000
AND DEPARTMENT_ID IN (60,90)
AND COMMISSION_PCT IS NULL;
```

Here you should know the priorities.
In this select statement, there are 2 conditions.
First condition: JOB_ID = 'AD_PRES' AND SALARY > 15000
Second condition: JOB_ID='SA_REP'

```
SELECT last_name, job_id, salary
FROM EMPLOYEES
WHERE JOB_ID = 'SA_REP'
OR JOB_ID = 'AD_PRES' AND SALARY > 15000;
```

Process: AND → OR

The following statement will give the same result.

```
SELECT last_name, job_id, salary
FROM EMPLOYEES
WHERE JOB_ID = 'SA_REP'
OR (JOB_ID = 'AD_PRES' AND SALARY > 15000);
```

Rules of Precedence

Operator	Meaning
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical operator
8	AND logical operator
9	OR logical operator

Note: You can use parentheses to override rules of precedence

Order by Clause

Order by

```
SELECT *
FROM EMPLOYEES
ORDER BY HIRE_DATE; -- the default order is always ASC: Ascending
```

Same result

```
SELECT *
FROM EMPLOYEES
ORDER BY HIRE_DATE ASC;
```

Order by DESC: Descending

```
SELECT * FROM EMPLOYEES
ORDER BY HIRE_DATE DESC
```

NULL values in order by

```
SELECT * FROM EMPLOYEES
ORDER BY COMMISSION_PCT; -- by default null values come "last" in Ascending order
```

```
SELECT * FROM EMPLOYEES
ORDER BY COMMISSION_PCT DESC; -- by default null values come "first" in Decending order
```

You can use NULLS FIRST to make null values appear first

```
SELECT *
FROM EMPLOYEES
ORDER BY COMMISSION_PCT NULLS FIRST;
```

You can use ORDER BY using column alias

```
SELECT FIRST_NAME N
FROM EMPLOYEES
ORDER BY N;
```

You can sort by expression

```
SELECT EMPLOYEE_IID, SALARY, SALARY+100
FROM EMPLOYEES
ORDER BY SALARY+100;
```

You can sort more than one columns

```
SELECT DEPARTMENT_ID, first_name, salary
FROM EMPLOYEES
ORDER BY DEPARTMENT_ID, FIRST_NAME;
```

DEPARTMENT_ID	FIRST_NAME
10	Jeniffer

20	Michael
20	Pat

```
SELECT DEPARTMENT_ID, first_name, salary
FROM EMPLOYEES
ORDER BY DEPARTMENT_ID ASC, FIRST_NAME DESC;
```

DEPARTMENT_ID	FIRST_NAME
10	Jeniffer
20	Pat
20	Michael

You can sort by column number in the select statement

```
SELECT DEPARTMENT_ID, first_name, salary
FROM EMPLOYEES
ORDER BY 1; -- 1 mean the first column in select statement (i.e. DEPARTMENT_ID)
```

The FETCH Clause

Using SQL Row Limiting Clause in a Query

```
SELECT employee_id, first_name
FROM EMPLOYEES
ORDER BY EMPLOYEE_ID;
```

```
SELECT employee_id, first_name
FROM EMPLOYEES
ORDER BY EMPLOYEE_ID
FETCH first 5 ROWS ONLY;
```

```
SELECT employee_id, first_name
FROM EMPLOYEES
ORDER BY EMPLOYEE_ID
FETCH first 50 PERCENT ROWS ONLY;
```

```
SELECT employee_id, first_name
FROM EMPLOYEES
ORDER BY EMPLOYEE_ID
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

```
SELECT employee_id, first_name
FROM EMPLOYEES
ORDER BY EMPLOYEE_ID
OFFSET 4 ROWS FETCH NEXT 50 PERCENT ROWS ONLY;
```

The meaning of TIES

```
SELECT EMPLOYEE_ID, first_name, salary
FROM EMPLOYEES
ORDER BY salary DESC;
```

```
SELECT EMPLOYEE_ID, first_name, salary
FROM EMPLOYEES
ORDER BY salary DESC
FETCH FIRST 2 ROWS ONLY;
```

```
SELECT EMPLOYEE_ID, first_name, salary
FROM EMPLOYEES
ORDER BY salary DESC
FETCH FIRST 2 ROWS WITH TIES;
```