



SQL Training



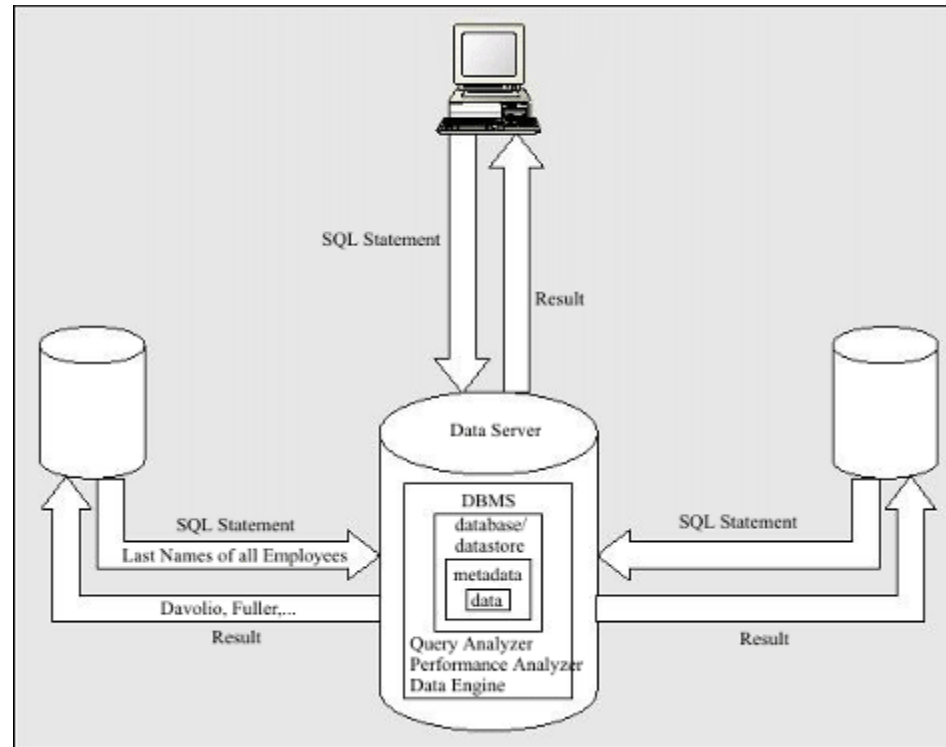


Objective

After completing this module, the participant will be able to do the following:

- **Define what is SQL**
- **Brief the history of SQL**
- **List the capabilities of SQL SELECT statements**
- **Execute a basic SELECT statement**
- **Limit the rows retrieved by a query**
- **Sort the rows retrieved by a query**

SQL is a standard computer language for accessing and manipulating databases.



SQL



- SQL stands for Structured Query Language
- SQL allows you to access a database
- SQL is an ANSI standard computer language
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert new records in a database
- SQL can delete records from a database
- SQL can update records in a database
- SQL is easy to learn

History



- The father of relational databases, and thus SQL, is Dr. E.F. "Ted" Codd who worked for IBM .
- SQL was adopted as a standard by
 - **ANSI** (American National Standards Institute) in 1986 and
 - **ISO** (International Organization for Standardization) in 1987.
- The SQL standard has gone through a number of revisions:

Year	Name	Alias
1986	SQL-86	SQL-87
1989	SQL-89	
1992	SQL-92	SQL2
1999	SQL:1999	SQL3
2003	SQL:2003	
2006	SQL:2006	

SQL Statements



SELECT	Data retrieval
INSERT UPDATE DELETE MERGE	Data manipulation language (DML)
CREATE ALTER DROP RENAME TRUNCATE	Data definition language (DDL)
COMMIT ROLLBACK SAVEPOINT	Transaction control
GRANT REVOKE	Data control language (DCL)

Capabilities Of SQL Select



Projection

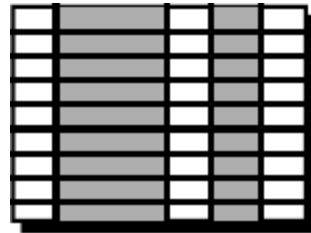


Table 1

Selection

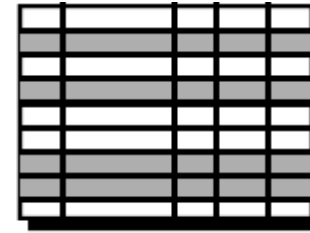


Table 1

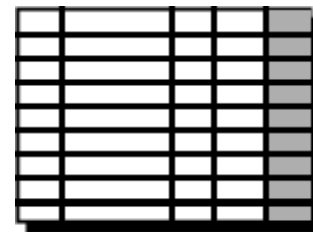


Table 1

Join

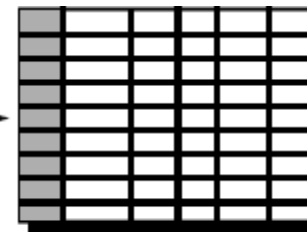


Table 2

SQL Select



Syntax

```
SELECT [DISTINCT|ALL ] { * | [columnExpression[AS  
newName]] [, ...] }  
FROM TableName [Aliase] [, ...] [WHERE  
condition]  
GROUP BY columnList [HAVING condition] [ORDER  
BY columnList]
```


Writing SQL Statements



- SQL statements are NOT case sensitive.
- SQL statements can be on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.

Projection Capability



- Used to choose the columns in a table that you want returned by your query.
- Can be used to choose as few or as many columns of the table as you require.

Examples:

```
SELECT Deptno, dname, loc  
FROM Dept;
```

```
SELECT *  
FROM Dept;
```

```
SELECT dname, Deptno  
FROM Dept;
```

```
SELECT DISTINCT Deptno  
FROM Emp;
```

Column Alias Name



- Renames a column heading

Examples:

```
SELECT Deptno Dept_no, dname Dept_Name, loc Location  
FROM Dept;
```

```
SELECT Deptno "Dept No", dname "DeptName", loc Location  
FROM Dept;
```

```
SELECT Deptno AS "Dept No", dname AS "Dept Name", loc  
AS Location FROM Dept
```

Arithmetic Operators



We can use arithmetic operators in any clause of a SQL statement except in the FROM clause.

Operators:

+ **-** ***** **/**

Example:

```
SELECT ename, sal, sal*12 FROM emp;
```

Operator Precedence :

***** **/** **+** **-**

Concatenation Operator And Literals



- **Concatenation Operator(||):** Concatenates columns or character strings to other columns
- **Literal:** A literal is a character, a number, or a date that is included in the SELECT list

Examples:

```
SELECT ename||sal AS " NAME And SALARY"  
FROM emp;
```

```
SELECT ename||' is earning Rs.'|| sal ||' per month' AS "  
NAME And SALARY"  
FROM emp
```

Selection Capability



- Used to choose the rows in a table that you want returned by a query.
- Various criteria can be used to restrict the rows that you see.
- Restrict the rows returned, by using the WHERE clause.

Syntax:

```
SELECT * | { [DISTINCT] column | expression [alias], ... }  
FROM table  
[WHERE condition(s)];
```

Operators Used IN Where Clause



- **Comparison :**
= , <> , < , > , <= , >=
- **Logical :**
AND , OR , NOT
- **Range:**
BETWEEN , NOT BETWEEN
- **Set Membership:**
IN , NOT IN
- **Pattern Match:**
LIKE , NOT LIKE
- **Null:**
IS NULL , IS NOT NULL

Comparison Search Condition



Conditions that compare one expression to another value or expression.

Examples:

```
SELECT *  
FROM emp  
WHERE deptno=20;
```

```
SELECT *  
FROM emp  
WHERE sal>1000 AND deptno=20
```

```
SELECT *  
FROM emp  
WHERE deptno = 20 OR Deptno=30
```


Range Search Condition



- **Range Condition:** You can display rows based on a range of values using the BETWEEN range condition. The range that you specify contains a lower limit and an upper limit

Examples:

```
SELECT ename, job, sal FROM  
emp  
WHERE sal BETWEEN 3000 AND 5000;  -(includes 3000 and 5000)
```

```
SELECT ename, job, sal FROM  
emp  
WHERE sal NOT BETWEEN 3000 AND 5000
```

Set Membership search Conditions



Set Membership

- Used to test for values in a specified set of values.
- Uses the keyword:
 - IN
 - NOT IN
- The membership condition is also known as IN condition.

Examples:

```
SELECT ename, hiredate, job FROM emp  
WHERE job IN ('MANAGER', 'CLERK', 'ANALYST');
```

```
SELECT ename, hiredate, job FROM emp  
WHERE job NOT IN ('MANAGER', 'CLERK', 'ANALYST');
```

Pattern Match Search Condition



- The pattern-matching operation is referred to as a **wildcard** search. Two symbols can be used to construct the search string.
- SQL has two special Pattern Matching symbols (wildcard)
 - %** - represents any sequence of zero or more characters
 - _** - represents any single character

Examples

```
SELECT ename,job,hiredate FROM emp
WHERE ename LIKE 'A%'
```

```
SELECT ename,job,hiredate
FROM emp
WHERE hiredate LIKE '%82';
```

```
SELECT ename,job,hiredate
FROM emp
WHERE job LIKE '_A%';
```

NULL Search Condition



- **NULL**
 - Means the value is Unavailable, unassigned, unknown, or inapplicable.
 - Cannot be tested with = because a null cannot be equal or unequal to any value.
 - Include the IS NULL condition and the IS NOT NULL condition.
 - IS NULL condition tests for nulls.

Examples:

```
SELECT ename, job, comm FROM  
emp  
WHERE comm IS NULL
```

```
SELECT ename, job, comm FROM  
emp  
WHERE comm IS NOT NULL
```

ORDER BY Clause



- Sort rows with the ORDER BY clause
 - ASC: ascending order, default
 - DESC: descending order
- The ORDER BY clause comes last in the SELECT statement.

- **Single column Ordering: Examples:**

```
SELECT ename, job, sal FROM emp  
ORDER BY sal;
```

```
SELECT ename, job, sal ,hiredate FROM emp  
ORDER BY hiredate DESC
```

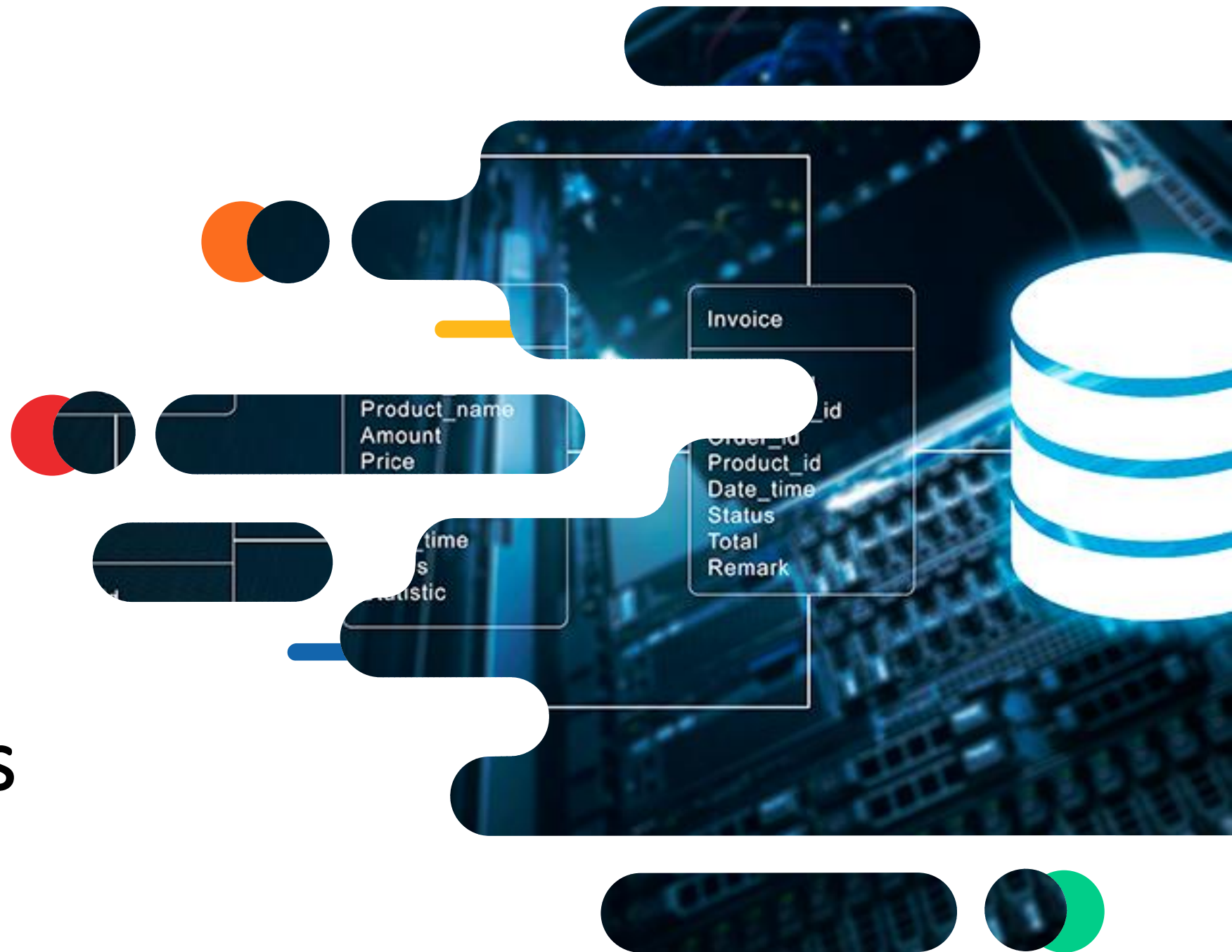
- **Multiple column Ordering: Examples:**

```
SELECT ename, job, deptno, sal,hiredate FROM emp ORDER BY deptno  
DESC, sal Asc
```

*Default Sort Order – Ascending



SQL Functions



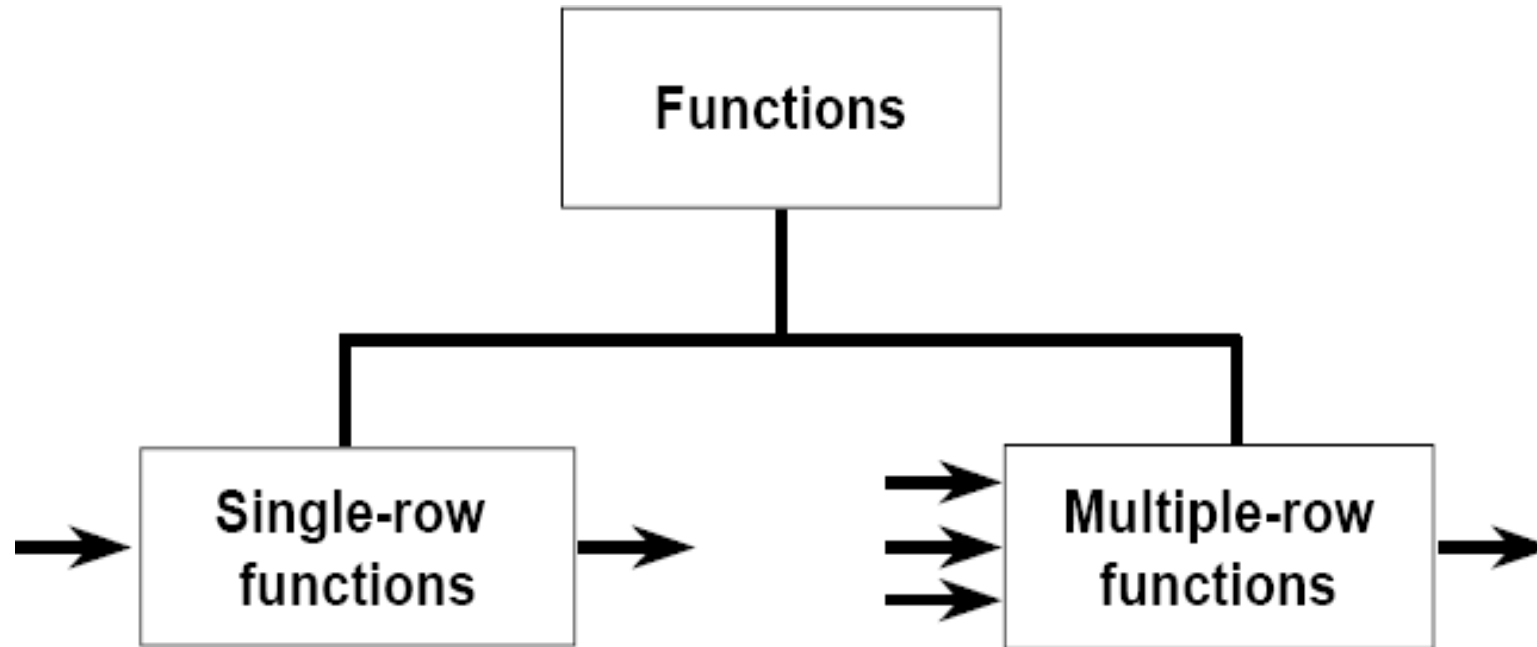


Objective

After completing this module, the participant will be able to do the following:

- **Describe various types of functions available in SQL**
- **Use character, number, and date functions in SELECT statements**
- **Describe the use of conversion functions**

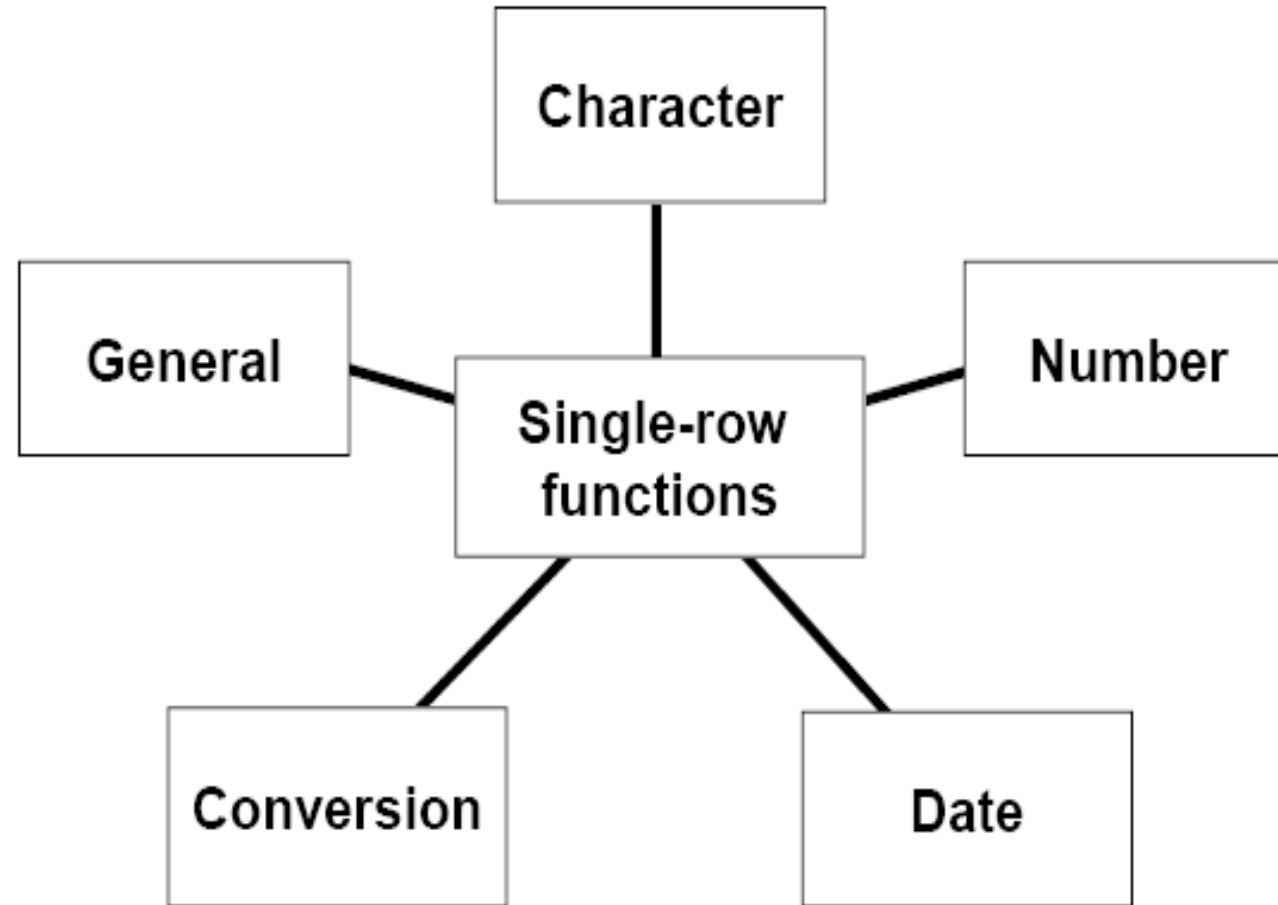
Types of SQL Functions



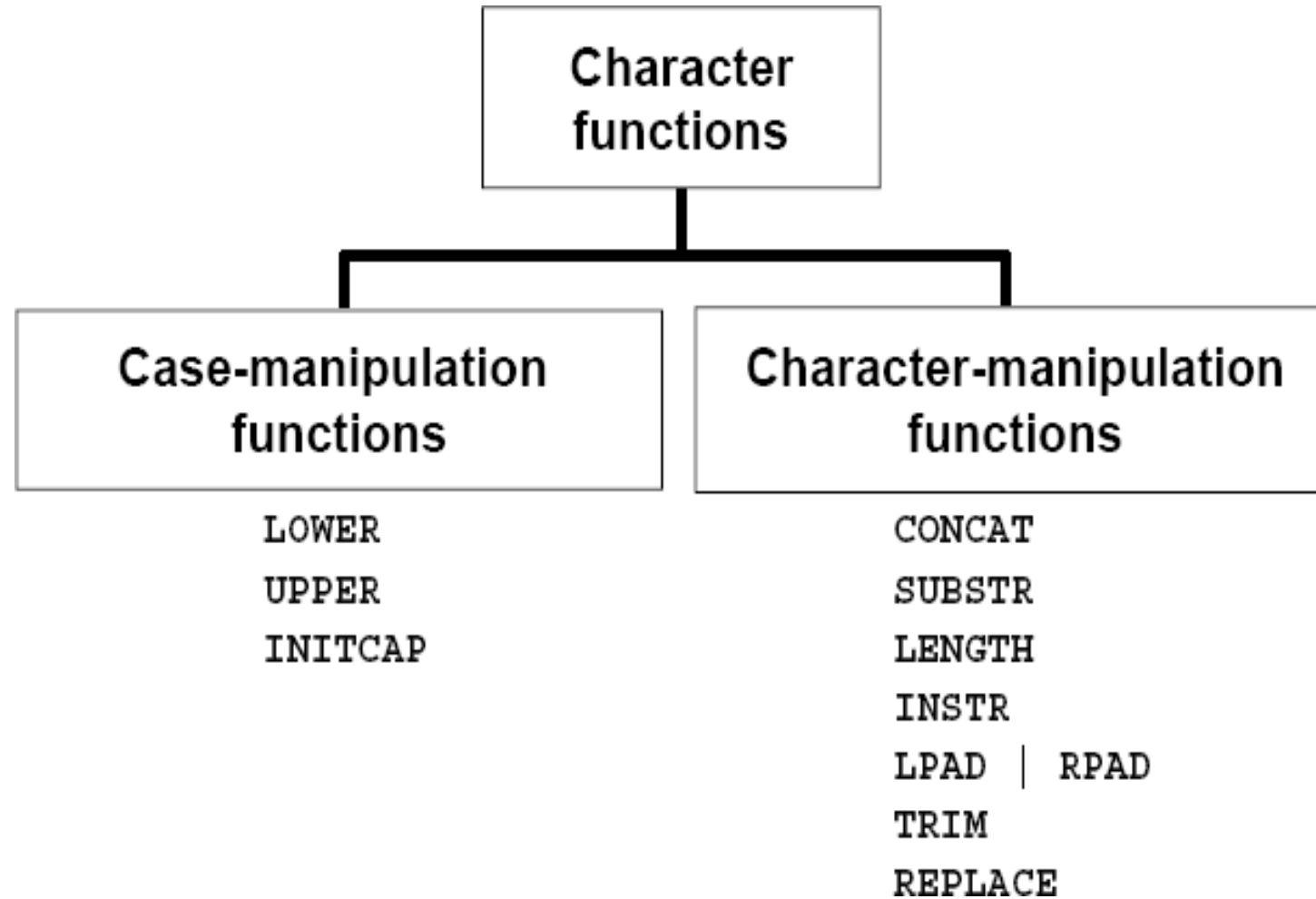
Single Row Functions



Single Row functions operate on single rows only and return one result per row.



Character Functions



Case Manipulation Functions



Function	Result
<code>LOWER('SQL Course')</code>	<code>sql course</code>
<code>UPPER('SQL Course')</code>	<code>SQL COURSE</code>
<code>INITCAP('SQL Course')</code>	<code>Sql Course</code>

Examples:

```
SELECT LOWER(ENAME) AS LOWER, UPPER(ENAME) AS  
UPPER, INITCAP(ENAME) AS INITCAP FROM emp;
```

```
SELECT LOWER('Sql') AS LOWER , UPPER('Sql') AS  
UPPER, INITCAP('sql') AS INITCAP FROM dual
```

Character Manipulation Functions



Examples:

```
select CONCAT(empno,ename) FROM emp;
```

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
TRIM('H' FROM 'HelloWorld')	elloWorld

Number Functions



- ROUND: Rounds value to specified decimal
`ROUND(45.926, 2)` = 45.93
- TRUNC: Truncates value to specified decimal
`TRUNC(45.926, 2)` = 45.92
- MOD: Returns remainder of division
`MOD(1600, 300)` = 100

Example:

```
SELECT ROUND (45.923, 2) , ROUND (45.923, 0) , ROUND (45.923, -1) FROM  
DUAL;
```

Date Functions



SYSDATE - function that returns current System date

Arithmetic with Dates

Operation	Result	Description
date + number	Date	Adds a number of days to a date
date - number	Date	Subtracts a number of days from a date
date - date	Number of days	Subtracts one date from another
date + number/24	Date	Adds a number of hours to a date

Examples

```
SELECT ename, (SYSDATE-hiredate)/7 AS WEEKS FROM emp;
```

Date Functions



Function	Description
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Next day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

Date Functions



- `MONTHS_BETWEEN ('01-SEP-95', '11-JAN-94')`
→ `19.6774194`
- `ADD_MONTHS ('11-JAN-94', 6)` → `'11-JUL-94'`
- `NEXT_DAY ('01-SEP-95', 'FRIDAY')`
→ `'08-SEP-95'`
- `LAST_DAY ('01-FEB-95')` → `'28-FEB-95'`

Date Functions



Assume SYSDATE = '25-JUL-95':

- `ROUND (SYSDATE, 'MONTH')` → 01-AUG-95
- `ROUND (SYSDATE , 'YEAR')` → 01-JAN-96
- `TRUNC (SYSDATE , 'MONTH')` → 01-JUL-95
- `TRUNC (SYSDATE , 'YEAR')` → 01-JAN-95

Conversion Functions



Syntax:

```
TO_CHAR(date, 'format_model');  
TO_CHAR(number, 'format_model');  
TO_NUMBER(char[, 'format_model'])  
TO_DATE(char[, 'format_model'])
```

Examples:

```
SELECT ename, TO_CHAR (hiredate, 'fmDD Month YYYY') AS  
HIREDATE  
FROM emp;
```

```
SELECT TO_CHAR(sal, '999,999.00') SAL FROM emp;
```

Conversion Functions



Elements of the Date Format Model

YYYY	Full year in numbers
YEAR	Year spelled out
MM	Two-digit value for month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

CASE Expression



Syntax:

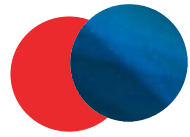
```
DECODE (col | expression, search1, result1  
[, search2, result2,...,] [,  
default])
```

Example:

```
SELECT last_name, job_id, salary,  
CASE job_id WHEN 'IT_PROG' THEN 1.10*salary WHEN  
'ST_CLERK' THEN 1.15*salary  
WHEN 'SA_REP' THEN 1.20*salary  
ELSE salary END "REVISED_SALARY"  
FROM employees;
```



Group Functions





Objective

After completing this module, the participant will be able to:

- Identify the available group functions
- Describe the use of group functions
- Group data using the GROUP BY clause
- Include or exclude grouped rows by using the HAVING clause

Group Functions



Group functions operate on sets of rows to give one result per group.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3600
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

...

20 rows selected.

The maximum
salary in
the **EMPLOYEES**
table.

MAX(SALARY)
24000

Group Functions



- COUNT
- SUM
- AVG
- MIN
- MAX

Group Functions



- COUNT

Examples:

```
SELECT COUNT(*) AS Count FROM emp;
```

```
SELECT COUNT(deptno) AS Count
```

```
FROM emp; SELECT COUNT(DISTINCT deptno) FROM emp;
```

COUNT(*) - Counts all rows of a table, regardless of whether nulls or duplicate values occur

Group Functions



- **SUM**

Examples:

```
SELECT SUM(sal) AS SALARY FROM emp;
```

```
SELECT COUNT(DISTINCT deptno) AS COUNT ,  
SUM(Sal) AS SALARY FROM emp  
WHERE job ='SALESMAN '
```

–You cannot use group functions in WHERE clause

Group Functions



- **MIN,MAX & AVG**

Examples:

```
SELECT MIN(sal) AS MIN_SAL,MAX(sal) AS MAX_SAL,AVG(sal) AS  
AVG_SAL FROM emp;
```

The GROUP BY Clause



Divide rows in a table into smaller groups

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

...

20 rows selected.

4400
9500
3500
6400
10033

The average salary in EMPLOYEES table for each department.

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

The GROUP BY Clause



Syntax

```
SELECT [column,] group_function(column), ...  
FROM table [WHERE condition]  
[GROUP BY column]  
[ORDER BY column];
```

Examples:

```
select count(empno) FROM emp GROUP BY deptno
```

```
SELECT deptno, COUNT(empno) AS COUNT, SUM(sal) AS SUM FROM emp GROUP BY  
deptno;
```

- All columns in the SELECT list that are not in group functions must be in the GROUP BY clause.
- The GROUP BY column does not have to be in the SELECT list.
- You cannot use a column alias in the GROUP BY clause.
- Using a WHERE clause, you can exclude rows before dividing them into groups.

Grouping by More Than One Column



Examples:

```
SELECT job,deptno,SUM(sal) AS salary FROM emp GROUP BY  
job,deptno
```

Restricting Groupings – Having Clause



EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
...	
20	6000
110	12000
110	6300

20 rows selected.

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

Restricting Groupings – Having Clause



In the same way that you use the WHERE clause to restrict the rows that you select, you use the HAVING clause to restrict Groups

Syntax

```
SELECT column, group_function
FROM table
[WHERE condition]
[GROUP BY group_by_expression] [HAVING
group_condition] [ORDER BY column];
```


Restricting Groupings – Having Clause



Example:

```
SELECT deptno, COUNT(empno) AS COUNT, SUM(sal) AS SUM FROM  
emp GROUP BY  
deptno HAVING SUM(sal)>9000
```



Thank you

Innovative Services



Passionate Employees

Delighted Customers

