Single Row Functions

Lecture 02

Agenda

- Character Functions
- Numeric Functions
- Datetime Functions
- Conversion Functions
- General Comparison Functions
- Assigning Variables

SQL Functions

- There are two types of functions:
  - Single-row
  - Multiple-row
- A single-row function returns one result for each row. These functions operate on single rows only and return one result for every row acted on.
- There are different types of Single-Row functions as follows:
  - Character
  - Number
  - Date
  - Conversion
  - General
- A multiple-row function returns one result per set of rows. Functions can manipulate groups of rows to give one result per group of rows. These functions are also called group functions.

Single-Row Functions

- Single-row functions return a single result for each row in the result set.
- Single-row functions can be used in
  - SELECT
  - WHERE
  - HAVING
  - ORDER BY
- Functions can be used to
  - Perform calculations on data
  - Modify individual data items
  - Manipulate output for groups of rows
  - Format dates and numbers for display
  - Convert column data types
- SQL functions may take arguments and always return a value.

Single-Row functions

- These functions manipulate data items.
  - Be a set to one or more arguments and return a single value for each row that is retrieved by the query.
  - An argument can be one of the following:
    - User supplied constant
    - Variable value
    - Column name
    - Expression
- The actions of single row functions include:
  - Acts on each row that is returned by the query
  - Returns one result per row
  - May possibly return a different data type than the one that is referenced
  - The function expects one or more arguments

Character Functions

Character Functions

- These functions accept character type arguments and return character or numeric values.

- Character functions:
  - Case manipulation
    - LOWER
    - UPPER
    - INITCAP
  - Character manipulation
    - SUBSTR
    - CONCAT
    - LENGTH
    - INSTR
    - TRIM
    - REPLACE

Case Manipulation Functions

Function	Returning result
LOWER('DATABASE Systems')	database systems
UPPER ('database sYstems')	DATABASE SYSTEMS
INITCAP('database Systems')	Database Systems

- LOWER()
  - returns the character argument with all lower case letters.
- UPPER()
  - returns the character argument with all upper case letters.
- INITCAP()
  - returns each world with the first letter capital and other letters lower case.
  - Words are delimited by white space or characters that are not alphanumeric.

Case Manipulation Functions (SQL Example 1)

```
SELECT LOWER(job_title) AS "Lower",
       UPPER(first_name) AS "upper",
       INITCAP(email) AS "Initcap"
FROM ass_employees
WHERE manager_id = 2;
```

	Lower	Upper	InitCap
1	accounting manager	ROSE	Rose.Stephens@Example.Com
2	administration assistant	ANNABELLE	Annabelle.Dunn@Example.Com
3	finance manager	MOHAMMAD	Mohammad.Peterson@Example.Com
4	human resources representative	HARPER	Harper.Spencer@Example.Com
5	public relations representative	GRACIE	Gracie.Gardner@Example.Com

Case Manipulation Functions (SQL Example 2)

- The following SQL query returns employees with “elli” word in their first name. To make sure you find all matching patterns, you can use **LOWER()** or **UPPER()** functions on the left side of the comparison expression and a matching pattern or word with all letters lower case of capital on the right hand side of the comparison expression.

```
select *
FROM employees
WHERE LOWER(first_name) LIKE 'elli%';
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE	HIRE_DATE	MANAGER_ID	JOB_TITLE
1	14	Elliot	Brooks	elliass_brooks@example.com	515.124.4567	07-DEC-16	9	Accountant
2	12	Elliott	James	elliott.james@example.com	515.124.4369	30-SEP-16	9	Accountant
3	88	Ellie	Robertson	ellie.robertson@example.com	650.509.4876	07-FEB-16	22	Shipping Clerk
4	31	Ellis	Washington	ellis.washington@example.com	650.124.6234	30-OCT-16	22	Stock Clerk

Character Manipulation Functions

Function	Returning result
CONCAT('Database', 'Systems')	DatabaseSystems
SUBSTR ('DatabaseSystems',1,4)	Data
LENGTH('DatabaseSystems')	15
INSTR('DatabaseSystems', 'b')	5
LPAD('Tommy', 10, '*')	*****Tommy
RPAD('Tommy', 10, '*')	Tommy*****
REPLACE('Jack and Jue', 'J', 'Bl')	Black and Blue
TRIM('D' FROM 'Database')	atabase
NVL(<column>, <replacement>)	Replaces NULL with <replacement> in <column>

Character Manipulation Functions (SQL Example 1)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	Sara
Jones	Cathy

```
SELECT CONCAT(fname, lname) AS "Full Name" FROM STAFF
```

Full Name
HarrySmith
LesKing
SaraMinor
CathyJones

```
SELECT CONCAT(fname, CONCAT(' ',lname)) AS "Full Name"
FROM STAFF
```

Full Name
Harry Smith
Les King
Sara Minor
Cathy Jones

Character Manipulation Functions (SQL Example 2)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	Sara
Jones	Cathy

```
SELECT SUBSTR(lname,1,3), SUBSTR(fname,2,2) FROM STAFF
```

1	2
Smi	ar
Kin	es
Min	ar
Jon	at

Character Manipulation Functions (SQL Example 3)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	Sara
Jones	Cathy

```
SELECT LENGTH(lname), LENGTH(fname) FROM STAFF
```

1	2
5	5
4	3
5	4
5	5

Character Manipulation Functions (SQL Example 4)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	Sara
Jones	Cathy

```
SELECT INSTR(lname,'o'), INSTR(fname,'a') FROM STAFF
```

1	2
0	2
0	0
4	2
2	2

Character Manipulation Functions (SQL Example 5)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	Sara
Jones	Cathy

```
SELECT LPAD(lname,10,'* '), RPAD(fname,10,'*') FROM STAFF
```

1	2
* * *Smith	Harry*****
* * * King	Les*******
* * *Minor	Sara******
* * *Jones	Cathy*****

Character Manipulation Functions (SQL Example 6)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	Sara
Jones	Cathy

```
SELECT REPLACE(lname,'o','*'), REPLACE(fname,'a','XX') FROM STAFF
```

1	2
Smith	HXXrry
King	Les
Min*r	SXXrXX
J*nes	CXXthy

Character Manipulation Functions (SQL Example 7)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	Sara
Jones	Cathy

```
SELECT TRIM(LEADING 'J' FROM lname), TRIM(TRAILING 'y' FROM fname),
TRIM(BOTH 's' FROM lname) FROM STAFF
```

1	2	3
Smith	Harr	Smith
King	Les	King
Minor	Sara	Minor
ones	Cath	Jone

Character Manipulation Functions (SQL Example 8)

STAFF

LNAME	FNAME
Smith	Harry
King	Les
Minor	-
Jones	-

```
SELECT LNAME, NVL(FNAME, 'N/A') FROM STAFF
```

LNAME	FNAME
Smith	Harry
King	Les
Minor	N/A
Jones	N/A

Numeric Functions

Numeric Functions

Function	Returning result
ROUND(5.678, 2)	5.68
TRUNC(5.678, 2)	5.67
MOD(10, 3)	1

- ROUND(*v, n*)
  - It receives two arguments.
    - *v:* is a value of any numeric data type.
    - *n*: is an integer value.
  - returns the argument value *v* rounded to *n* places to the right of the decimal point.
  -

ROUND()

- ROUND(*v, n*)
  - It receives two arguments.
    - *v:* is a value of any numeric data type.
    - *n*: is an integer value.
  - returns the argument value *v* rounded to *n* places to the right of the decimal point.
  - The ROUND() function with out the second parameter *n*, the default value 0 will be considered for the second argument.
  - If you use 0 or no value for the second argument, *n* is rounded to zero decimal places.
    - ROUND(10.96) → 11

TRUNC()

- TRUNC(v, n)
  - truncates a number *n* to *v* decimal places.
  - TRUNC(15.193, 2) → 15.19
  - TRUNC(15.193, 3) → 15
  - TRUNC(15.193, 1) →15.1

- TRUNC(n)
  - truncate a number *n* to zero decimal places.
  - TRUNC(15.193) → 15

MOD()

- MOD(*v*, *n*)
  - *n*: dividend
  - *v*: divider
  - The function MOD() returns the remainder of the division of *n* and *v*.
- MOD(121,14) → 9
- MOD(25, 7) → 4
- The MOD() function is used to determine if a number is *odd* or *even*.

Numeric Function Example

```
SELECT product_id, MOD(product_id,7),
list_price, ROUND(list_price,1) , ROUND(list_price),
TRUNC(list_price,1), TRUNC(list_price)
FROM ass_products
WHERE list_price < 50
ORDER BY product_id;
```

	PRODUCT_ID	MOD(PRODUCT_ID,7)	LIST_PRICE	ROUND(LIST_PRICE,1)	ROUND(LIST_PRICE)	TRUNC(LIST_PRICE,1)	TRUNC(LIST_PRICE)
1	44	2	49.37	49.4	49	49.3	49
2	56	0	16.99	17	17	16.9	16
3	94	3	15.55	15.6	16	15.5	15
4	168	0	43.99	44	44	43.9	43
5	235	4	41.99	42	42	41.9	41
6	256	4	26.99	27	27	26.9	26
7	268	2	47.88	47.9	48	47.8	47

Datetime Functions

Dates

- The default display and input format for any date is DD-MON-RR. Valid Oracle dates are between January 1, 4712 B.C., and December 31, 9999 A.D.
- In the example in the slide, the HIRE_DATE column output is displayed in the default format DD-MON-RR. However, dates are not stored in the database in this format. All the components of the date and time are stored. So, although a HIRE_DATE such as 17-JUN-87 is displayed as day, month, and year, there is also time and century information associated with the date. The complete data might be June 17, 1987, 5:10:43 p.m.
  - CENTURY YEAR MONTH DAY HOUR MINUTE SECOND
  - 19 87 06 17 17 10 43
- Note: century or year stored as 4 digits even if displayed as 2

SYSDATE

- SYSDATE returns current
  - Date
  - Time

```
SELECT SYSDATE
FROM dual;
```

	SYSDATE
1	04-MAR-20

Arithmetic Operations on Dates

- Since Oracle database stores dates as numbers, arithmetic operations such as addition or subtraction can be performed on date values. You can add or subtract both numbers and dates to or from date values.
  - Date + Number
  - Date – Number
  - Date – Date
- Example:
  - The next billing due date is calculated which 30 days from today.

```
SELECT sysdate, sysdate + 30
FROM dual;
```

	SYSDATE	SYSDATE+30
1	04-MAR-20	03-APR-20

Dates and Arithmetic Operators

- Example: Find how many weeks an employee has worked at the company?

```
SELECT   last_name, (sysdate - hire_date)/7 "Weeks Employed" -- returns days convert to weeks
FROM     employees
WHERE    department_id = 90;
```

	LAST_NAME	Weeks Employed
1	Onopriyenko	971.9412202380952380952380952380952380952380095
2	King	1707.0840773809523809523809523809523809523809
3	Kochhar	1588.94122023809523809523809523809523809523
4	De Haan	1416.0840773809523809523809523809523809523809

- To improve the result format, we use function TRUNC()

```
SELECT   last_name, TRUNC((sysdate - hire_date)/7,2) "Weeks Employed"
FROM     employees
WHERE    department_id = 90;
```

	LAST_NAME	Weeks Employed
1	Onopriyenko	971.94
2	King	1707.08
3	Kochhar	1588.94
4	De Haan	1416.08

Dates and Arithmetic Operators

https://www.db2tutorial.com/db2-date-functions/

- Now lets take a look at a selection of the functions

Conversion Functions

Conversion Functions

- Conversions are about changing the data type of a column.
- This may be required to bridge between one SQL statement and another – or – to bridge between the way an application works versus the way a database is designed (for example: application performs an arithmetic operation on numerical values stored as a CHAR or VARCHAR in the datebase)
- This cannot be performed in an ALTER TABLE
- This is about changing the data type in a result set
- Some data types allow for conversions, some conditionally allow for conversions and some do not.
  - An example of a valid data type conversion, which will always work: INTEGER to CHAR/VARCHAR
  - An example of a valid data type conversion, which may not work: CHAR/VARCHAR to INTEGER
  - An example of a valid data type conversion, which may produce a warning: VARCHAR to CHAR
  - An example of an invalid data type conversion: INTEGER to BINARY

Conversion Functions

- Conversions are primarily performed through explicitly executing the CAST command

CREATE TABLE EMPLOYEE (EMPNO INTEGER, SALARY DECIMAL)

SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE;

- In some cases, the database manager will automatically perform a cast under the covers

CREATE TABLE EMPLOYEE (EMPNO INTEGER, SALARY DECIMAL, PHONE INTEGER)

SELECT EMPNO, SUBSTR(PHONE, 1, 3) FROM EMPLOYEE

Conversion Functions

Cast from data type –	To data type[1]																				
	SMALLINT	INTEGER	BIGINT	DECIMAL	DECFLOAT	REAL	DOUBLE	CHAR	VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	DBCLOB	BINARY	VARBINARY	BLOB	DATE	TIME	TIMESTAMP WITHOUT TIME ZONE	TIMESTAMP WITH TIME ZONE	ROWID	XML
SMALLINT	Y	Y	Y	Y	Y	Y	Y	Y	Y													
INTEGER	Y	Y	Y	Y	Y	Y	Y	Y	Y													
BIGINT	Y	Y	Y	Y	Y	Y	Y	Y	Y													
DECIMAL	Y	Y	Y	Y	Y	Y	Y	Y	Y													
DECFLOAT	Y	Y	Y	Y	Y	Y	Y	Y	Y													
REAL	Y	Y	Y	Y	Y	Y	Y	Y	Y													
DOUBLE	Y	Y	Y	Y	Y	Y	Y	Y	Y													
CHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
VARCHAR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
CLOB								Y	Y	Y	Y	Y	Y	Y	Y	Y						
GRAPHIC	Y	Y	Y	Y	Y	Y	Y	Y[2]	Y[2]	Y[2]	Y	Y	Y	Y	Y	Y	Y[3]	Y[3]	Y[3]	Y[3]		
VARGRAPHIC	Y	Y	Y	Y	Y	Y	Y	Y[2]	Y[2]	Y[2]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y[3]		
DBCLOB								Y[2]	Y[2]	Y[2]	Y	Y	Y	Y	Y	Y						
BINARY														Y	Y	Y						
VARBINARY														Y	Y	Y						
BLOB														Y	Y	Y						
DATE								Y	Y								Y					
TIME								Y	Y									Y				
TIMESTAMP WITHOUT TIME ZONE								Y	Y								Y	Y	Y	Y		
TIMESTAMP WITH TIME ZONE								Y	Y								Y	Y	Y	Y		
ROWID								Y	Y					Y	Y	Y					Y	
XML																						Y

General Comparison Functions

Comparison Functions

- Comparison functions allow for two or more values to be compared during a SQL statement to help define the desired result set. There is a large set of comparison functions.

- Comparison Functions:
  - >, <, <=, >=, =, !=, <>
  - AND
  - OR
  - IN
  - NOT IN
  - BETWEEN
  - NOT BETWEEN
  - LIKE
  - NOT LIKE
  - IS
  - IS NOT

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY > 70000

LNAME	FNAME	SALARY
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY >= 70000

LNAME	FNAME	SALARY
Minor	Sara	70,000
Jones	Cathy	80,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY < 70000

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000

SELECT * FROM STAFF WHERE SALARY <= 50000

LNAME	FNAME	SALARY
Smith	Harry	50,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY = 60000

LNAME	FNAME	SALARY
King	Les	60,000

SELECT * FROM STAFF WHERE SALARY <> 50000

LNAME	FNAME	SALARY
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY < 60000 OR SALARY > 70000

LNAME	FNAME	SALARY
Smith	Harry	50,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY > 50000 AND LNAME = 'King'

LNAME	FNAME	SALARY
King	Les	60,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE FNAME IN ('Les', 'Sara')

LNAME	FNAME	SALARY
King	Les	60,000
Minor	Sara	70,000

SELECT * FROM STAFF WHERE FNAME NOT IN ('Les', 'Sara')

LNAME	FNAME	SALARY
Smith	Harry	50,000
Jones	Cathy	80,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY IN (60000, 70000)

LNAME	FNAME	SALARY
King	Les	60,000
Minor	Sara	70,000

SELECT * FROM STAFF WHERE SALARY NOT IN (60000, 70000)

LNAME	FNAME	SALARY
Smith	Harry	50,000
Jones	Cathy	80,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE SALARY BETWEEN 55000 AND 75000

LNAME	FNAME	SALARY
King	Les	60,000
Minor	Sara	70,000

SELECT * FROM STAFF WHERE SALARY NOT BETWEEN 55000 AND 75000

LNAME	FNAME	SALARY
Smith	Harry	50,000
Jones	Cathy	80,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE FNAME LIKE '%a%'

LNAME	FNAME	SALARY
Smith	Harry	50,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE FNAME NOT LIKE '%a%'

LNAME	FNAME	SALARY
King	Les	60,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE FNAME LIKE '%ar%'

LNAME	FNAME	SALARY
Smith	Harry	50,000
Minor	Sara	70,000

SELECT * FROM STAFF WHERE FNAME LIKE '%y'

LNAME	FNAME	SALARY
Smith	Harry	50,000
Jones	Cathy	80,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

SELECT * FROM STAFF WHERE FNAME LIKE '%ar%y'

LNAME	FNAME	SALARY
Smith	Harry	50,000

SELECT * FROM STAFF WHERE FNAME LIKE 'C%y'

LNAME	FNAME	SALARY
Jones	Cathy	80,000

Comparison Functions

STAFF

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000
Thomson	-	90,000

SELECT * FROM STAFF WHERE FNAME IS NULL

LNAME	FNAME	SALARY
Thomson	-	90,000

SELECT * FROM STAFF WHERE FNAME IS NOT NULL

LNAME	FNAME	SALARY
Smith	Harry	50,000
King	Les	60,000
Minor	Sara	70,000
Jones	Cathy	80,000

Assigning Variables

Assigning Variables

- Many database systems allow you to assign variables then use those variables in SQL statements
- Quite often you'll find slightly different syntax between database systems on how this is handled:
- Db2 Example:
  - CREATE VARIABLE empnum INTEGER
  - SET empnum = 20
  - SELECT * FROM STAFF WHERE id = empnum
- Oracle Example:
  - DEFINE emp_id NUMBER = 107;
  - SELECT * FROM EMPLOYEES WHERE employee_id = &emp_id;
  - UNDEFINE emp_id;