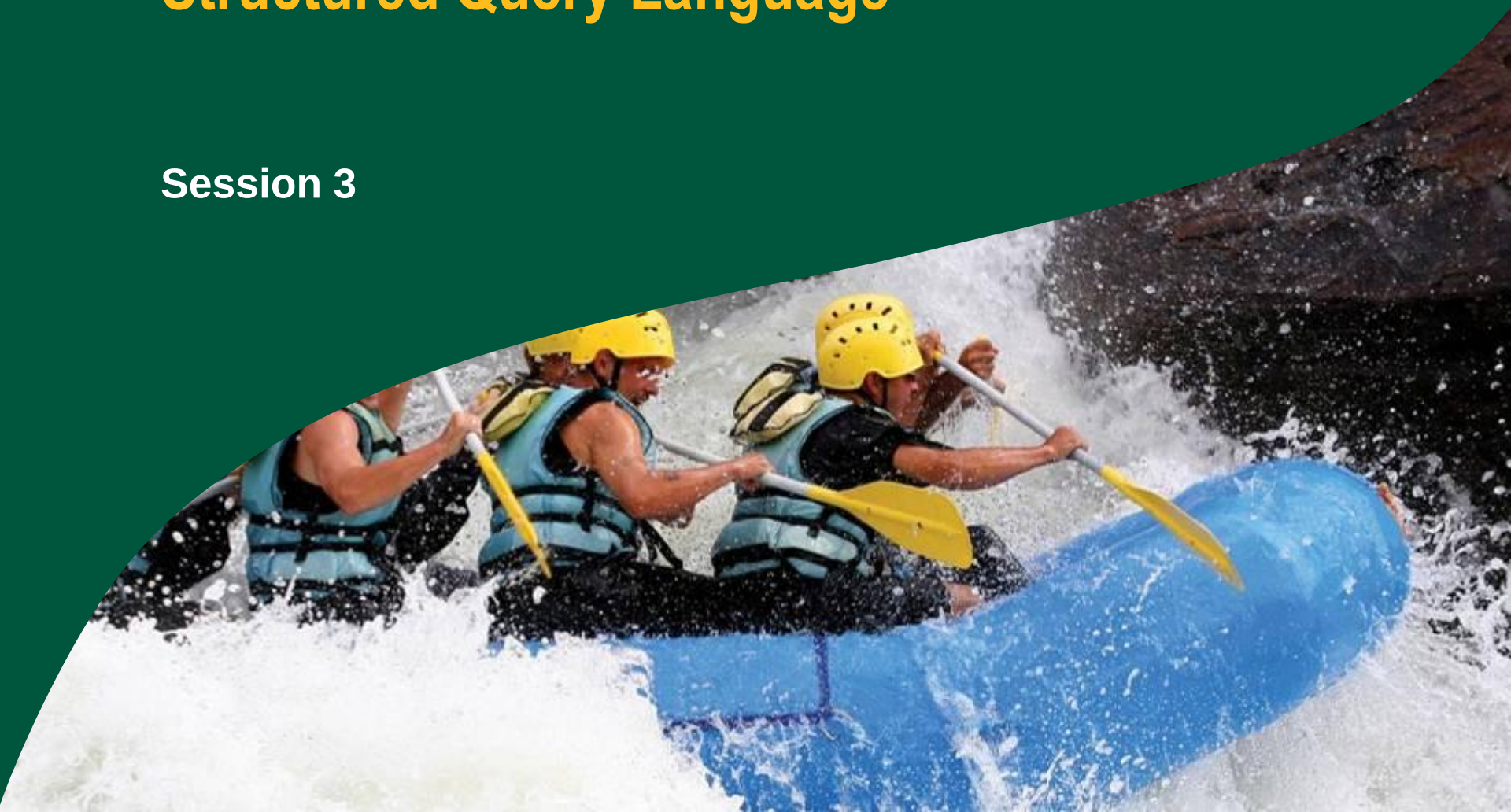
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Structured Query Language

Session 3

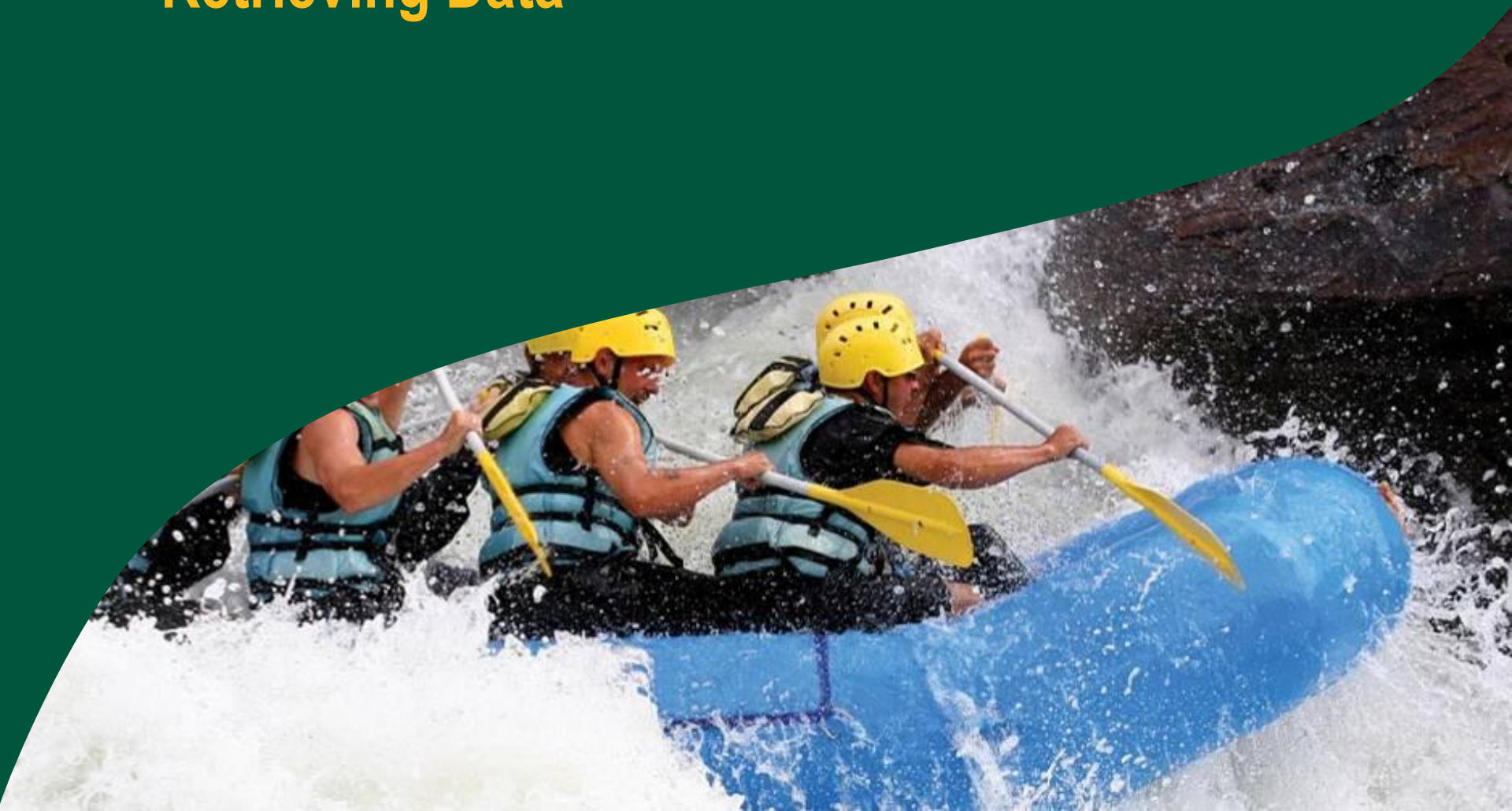


Coverage

- Retrieving Data
- Ordering Data
- Substitution Variables

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Retrieving Data



Retrieving data

Select Distinct /*/<column_list>

From <table_name>

Where <conditions>

Group By <column_list>

Having <conditions>

Order By <column_list>

Simple retrieval examples

```
Select * from emp;

Select empno,ename,deptno,sal
From emp;

Select dname from dept;

Select empno,ename deptno, sal from emp
where empno = 7900;
```

Performing Arithmetic Calculation on Columns

- Airthmetic Opeartors **+ , * , / , -** can be used on numeric or date datatype columns

Select empno,ename,sal, **sal*12** from emp;

Select ename,hiredate, **hiredate + 30** from emp;

Select ename,hiredate, **sysdate - hiredate** from emp;

Select empno,ename,sal,comm, **sal + comm** from emp;

Operator Precedence

- * / + -
- Multiplication and division take priority over addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to force prioritized evaluation and to clarify statements

Handling NULL Values

Select empno,ename,sal,comm,sal + comm from emp;

- Anything manipulated by NULL value will always be NULL
- NULL values can be handled by using NVL function
- **NVL(<column_name/value>,<value_substituted>)**
- NVL function helps to substitute NULL value with a different value of the same datatype

Select empno,ename,sal,comm,sal + nvl(comm,0) from emp

Concatenating Columns / Literals

- || (Pipe) is used to join two columns or strings of same or different data-type
- Resultant column is of character datatype

```
select ename || job from emp;
select ename || '  - ' || job from emp;
select 'Employee ' || ename || ' is working since ' ||  hiredate
  from emp;
```

Giving Alias name to Columns

```
Select empno,ename,sal,sal*12 annual_salary from emp;

Select empno,ename,sal,sal*12 AS annual_salary from
  emp;

Select empno,ename,sal,comm "Commission Earned"
  from emp;

Select 'Employee ' || ename || ' is working since ' ||  hiredate
  Employee_Information from emp;
```

Eliminating Duplicate Rows

- **Distinct** displays unique values for the columns

select **distinct** job from emp;

select **distinct deptno , job** from emp;

- Distinct must be the first keyword after Select
- Must be mentioned only once in the select clause
- The values of all the columns mentioned in the select list are taken into consideration

Relational Operators

=	Checks for equality comparison between two values
<> , != , ^=	Checks for in-equality comparison between two values
>	Checks for greater than value
<	Checks for less than value
>=	Checks for greater than and equal to value
<=	Checks for less than and equal to value
[NOT] Between…And	Checks for value in Between the Range of Values , inclusive at both ends
[NOT] In	Checks for specific list of values. Also known as membership condition
[NOT] Like	Checks for matching pattern using wildcards - % for all characters and _ for a single character. Self Study : Use ESCAPE option to search for % and _]
Is [NOT] NULL	Checks for NULL value

Logical Operators

AND	Used to join two conditions with AND logical operator. Both of the conditions should evaluate to TRUE
OR	Used to join two conditions with OR logical operator. Either of the condition can be TRUE or FALSE
NOT	Used for negating the result of the condition

Examples

```
Select * from emp
where deptno = 10;
```

```
Select * from emp
where sal > 3000;
```

```
Select * from emp
where deptno in (10,30);
```

Examples

```
Select * from emp
where sal between 1000 and 3000;

Select * from emp
where hiredate between '01-JAN-85' and '01-JAN-95';

Select * from emp
where comm is not null;

Select * from emp
where ename like 'B%';
```

Examples

```
select * from emp
where ename like '_L%';

select * from emp
where sal > 2000
and deptno = 10;

select ename,sal,comm,sal+nvl(comm,0) from emp
 where sal > 1500
OR sal+nvl(comm,0) > 1500;

select * from emp
where not sal > 2000;
```

Rules of Precedence

Order Evaluated	Operator
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT logical condition
7	AND logical condition
8	OR logical condition

Use parentheses to Override rules of precedence

And Truth table

And	True	False	Null
True	True	False	Null
False	False	False	False
Null	Null	False	Null

Or Truth table

OR	True	False	Null
True	True	True	True
False	True	False	Null
Null	True	Null	Null

NOT Truth table

	True	False	Null
NOT	False	True	Null

Ordering data

- Data is randomly retrieved in any order.
- ORDER BY can be used to order the display of data in a specific order
- Last clause in the Select statement
- **ORDER BY <column> ASC/DESC , <column> ASC/DESC ,.**

```
select * from emp
order by ename;

select * from emp
order by deptno,ename;
```

Order By….

- Calculated columns can be used to order data
- Column alias names can be used to order data
- Columns not included in the select list can be used to order data
- Null values appear last in an ascending sort and first in a descending sort
- Column can be sorted by column number in the select list

```
Select ename,sal,comm, SAL+NVL(COMM,0)
 from emp
 where sal > 1500
order by  SAL+NVL(COMM,0)

Select ename,sal,comm, SAL+NVL(COMM,0)  total_salary
 from emp
 where sal > 1500
order by  total_salary
```

Substitution Variables

- In SQL * PLUS substitution variables are used to temporarily store values.

- Substitution variable can be created as –
  - Defining user variable with DEFINE
  - Substitution method by using & and &&

Define

- Use Define to declare variables and assign values to them
- Define variable = value

  will create variable and assign it a value
- Define variable

  will display variable its value and data type.
- Define

  will display all user variables their data type and value
- Undefine command will remove the variable.

Using &

- Runtime variables are created by using ‘&’ , their scope is only in the SQL statement in which they are defined.
- Every time query is executed , user have to give value for this variable.

e.g.

```
Select * from emp
Where deptno = &deptno;
```

Using &&

- When value of the variable is to be reused

  i.e. user should not be prompted to enter value for every execution of query.

  '&&' is used.

e.g. Select **&&deptno, dname** from emp

Where deptno = **&deptno;**

Table Alias Name

- An alias name which is a short name can be give to a table by specifying it after the table name.
- The alias name and column name can be used as short – cuts to table name and column name specification.

```
Select empno, ename, deptno
From emp e;

Select e.empno, e.ename, e.deptno
From emp e;
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Thank You !!!