SQL – Part 2

Data Manipulation Language (DML)

Other SQL functionalities:

- ► Queries
- ► Operations
- ► Ordering the results

Query

Based on the given tables, produce a list of job grades showing the minimum and maximum salary for each grade.

Tables:
employees(id, First_name, surname, salary, dept_id, comm_pct)
departments(dept_id, dept_name, manager_id)
jobgrades(grade_level,min_sal,max_sal)

Answer:

```
SELECT grade_level,min_sal, max_sal FROM jobgrades;
```

Or (*because we want ALL the attributes*)

```
SELECT* FROM jobgrades;
```

SELECT …….

Create a query to display all staff ids, their first names and surnames. Decide table(s) required and columns from that table(s) to create query.

Answer:

SELECT emp_id, first_name, surname FROM employees;

Output

- Headings are listed in the order they appear on selectlist.
- Names on selectlist must be spelt exactly as they appear in the named table.
- When "*" is used, headings are in the order of the attributes in the table structure.

Headings

- Some names in the headings of list produced are not user friendly:

  ➔ emp_id

- For example, the given requirement is as follows:

  The list should be headed as:

  ID First Name Last Name

Renaming the headings

- Renames a column heading.
- Follows the column name immediately can be preceded by AS.
- Must be surrounded by double quotation marks if it contains spaces or special characters or it is case-sensitive.
- Useful with arithmetic expressions or calculations.

Renaming…….

SELECT surname , **first_name AS "First Name"**

FROM employees;

List will be headed Surname **First Name**

SELECT and WHERE

- For limiting rows returned restricts rows to those that meet a condition.

Syntax:

```
SELECT selectlist FROM tablename
WHERE logical expression(s);
```

Operations

*	Multiply
/	Divide
+	Add
-	Subtract

Operation.....

Symbol	Operator
=	Equal
>	Greater than
>=	Greater than or Equal to
<	Less than
<=	Less than or equal to
Between.. AND..	Between two values (Inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a NULL value
!=	Not Equal

Operation…..

AND

True if its two component conditions are true

OR

True if either condition is true

NOT

True if condition is false

Examples

- Produce a list of employees in department mk01. The list should show each employee's surname, employee id and departmental code.

Answer:

SELECT surname, emp_id AS "Employee ID", dept_id AS "Departmental Code" FROM employees WHERE dept_id = 'mk01';

More Examples

- List of all customer details with CustiD of 10 in table Customers.

```
SELECT* FROM Customers WHERE CustiD=10;
```

- List grade level and max salary where min salary is greater than 35000 in jobgrades table.

```
SELECT grade_level AS "Grade", max_sal AS "Max
Salary" FROM jobgrades
WHERE min_sal> 35000;
```

More Examples…

- List from Customers table those customers who are from UK and Coventry

  SELECT* FROM Customers WHERE Country='UK‘ AND City='Coventry';

- List customers who are from UK and (Coventry or Sunderland)

  SELECT * FROM Customers WHERE Country='UK' AND (City='Coventry'
  OR City='Sunderland');

Note:

→ UK and Coventry (from the example above) must be enclosed in a pair of single quotes.

→ Case sensitive

Comparison: 'Smith' is not the same as 'SMITH'

Examples....

- Display the ID, Department and salary of employer(s) whose surname(s) is (are) Smith:

Answer:

```
SELECT emp_id AS "ID", dept_id AS "Department", salary
FROM employees
WHERE surname= 'Smith';
```

Examples….

- **LOWER** (Column/expression)

  Converts value into lower case letters
- **UPPER** (Column/expression)

  Converts into upper case letters
- **INITAP** (Column/expression)

  Converts first character (left most) into upper case

Example:

```
SELECT emp_id AS "10", UPPER(first_name) AS "First
Name", salary
FROM employees;
```

Ordering the results!

Syntax:

```
SELECT selectlist
FROM tablename
[WHERE conditional expression(s)]
[ORDER BY {col1, expr, position_of_col}
[ASC|DESC]];
```

ASC or DESC

ASC sorts rows in ascending order (default)

DESC sorts rows in descending order

Examples….

- Produce a list of staff in ascending order of salary; the list should include staff id, surname and salary.

Answer:

```
SELECT emp_id AS "staff id", surname AS "Surname",
salary FROM employees
ORDER BY salary ASC;
```

ORDER BY- Example

- Select all records from a Customer table and order by Country and CustomerName attributes:

```
SELECT* FROM Customers
ORDER BY Country, CustomerName;
```

- Select all emp_id, first_name and surname from a Customer table and order by Surname (A-Z), first_name (Z-A)

```
SELECT emp_id, first_name, surname
FROM employees
ORDER BY surname ASC, first_name  DESC;
```

Updating a row

Syntax:

```
UPDATE tablename
SET column1 = value1,[column2 = value2,...]
[WHERE condition];
```

Example:

Change Mr King’s salary to £25000.

```
UPDATE employees
SET salary= 25000
WHERE surname = "King";
```

What would happen here?

```
UPDATE employees
SET salary= 25000;
```

Can I specify a range? Yes, you can!

- Range has a lower limit and an upper limit.
- Lower limit must be specified first.
- Values specified are inclusive.

Examples

- Produce a list of employees that earn salaries between 30000 and 45000.

```
SELECT surname, salary FROM employees
WHERE salary BETWEEN 30000 AND 45000;
```

<u>Other examples that specifying a range:</u>

```
SELECT first_name AS "First Name",surname
FROM employees WHERE surname BETWEEN 'john' AND ‘peter' ;
```

```
SELECT* FROM Orders WHERE OrderDate BETWEEN '01-jan-
2014' AND '07-jan-2014';
```

Summary

Relational database functions, also known as:

CRUD

- Create
- Read
- Update
- Delete