

## LAB/ PRACTICAL

### DATA MANIPULATION LANGUAGE - SELECTION AND PROJECTION

#### LEARNING OUTCOME

By the end of the lesson the student will be able to learn about :

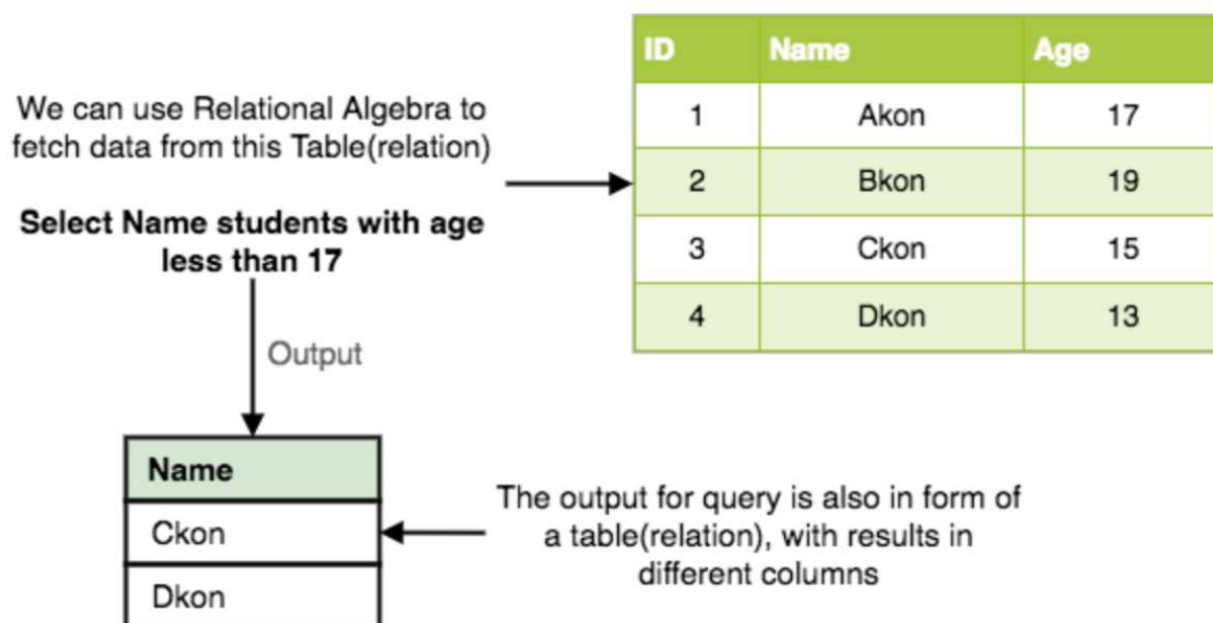
Understand and know what is Relational Algebra (RA) for Selection and Projection

Transform from RA to SQL or vice versa

#### What is Relational Algebra?

It uses procedural query language. It collects instances of relations as input and gives occurrences of relations as output. It uses various operations to perform this action. Relational algebra operations are performed recursively on a relation. The output of these operations is a new relation, which might be formed from one or more input relations.

In relational algebra, input is a relation(table from which data has to be accessed) and output is also a relation(a temporary table holding the data asked for by the user).



(source: <https://www.studytonight.com/dbms/relational-algebra.php>)

#### Select Operation ( $\sigma$ )

The sigma symbol is used to fetch rows(tuples) from table(relation) which satisfies a given condition.

**Syntax:**

$\sigma_p(r)$

where:

$\sigma$  - Select predicate

r - the name of relation (table name in which we want to look for data)

p - is the propositional logic, where we specify the conditions that must be satisfied by the data

(Note: In propositional logic, one can use unary and binary operators like =, >, <, etc, to specify the conditions.

Example: Fetch data for students with age more than 17 from table (relation) Student can be written in RA as below:

$\sigma_{age > 17}(\text{Student})$  --> can also be pronounced as  $\sigma_{age > 17}(\text{Student})$

The SQL for this statement is:

**SELECT \* FROM Student WHERE age > 17**

In order to specify two conditions (or more) we can use AND, OR, or other logical operators.

Example: Fetch Student with information of male students (from field gender), of age more than 17 can be written in RA as below:

```
age > 17 ^ gender='Male' (Student) --> sigma age > 17 ^ gender='Male' (Student)
```

The SQL for this statement is:

```
SELECT * FROM Student WHERE age=17 AND gender='Male'
```

### Project Operation ( $\Pi$ )

Project operation is used to project only a certain set of attributes of a relation. In simple words, if you want to see only the names of all of the students in the Student table, then you can use **Project Operation**.

It will only project or show the columns or attributes you asked for, and will also remove duplicate data from the columns. Syntax:

```
 $\Pi_{A1, A2...}(r)$ 
```

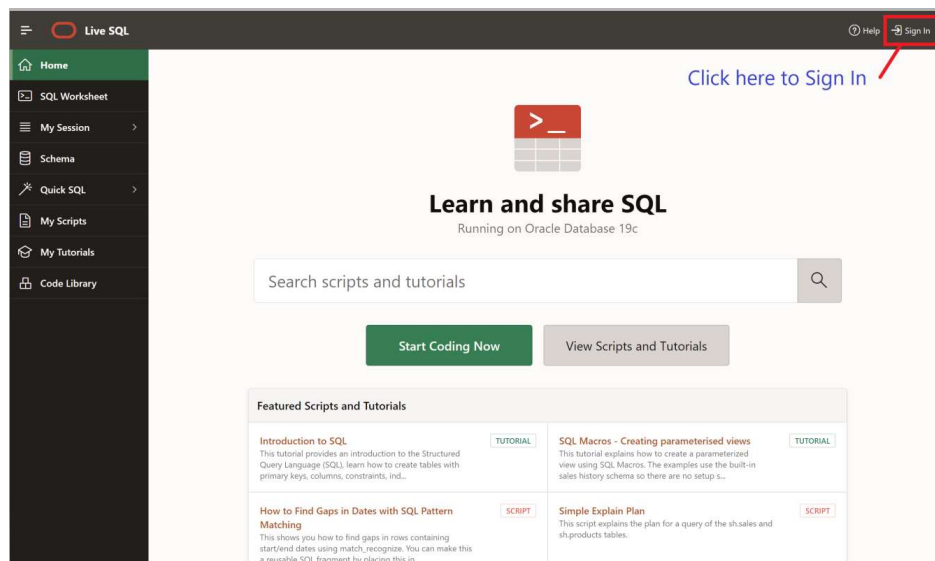
where

A1, A2 etc are attribute names (column names).

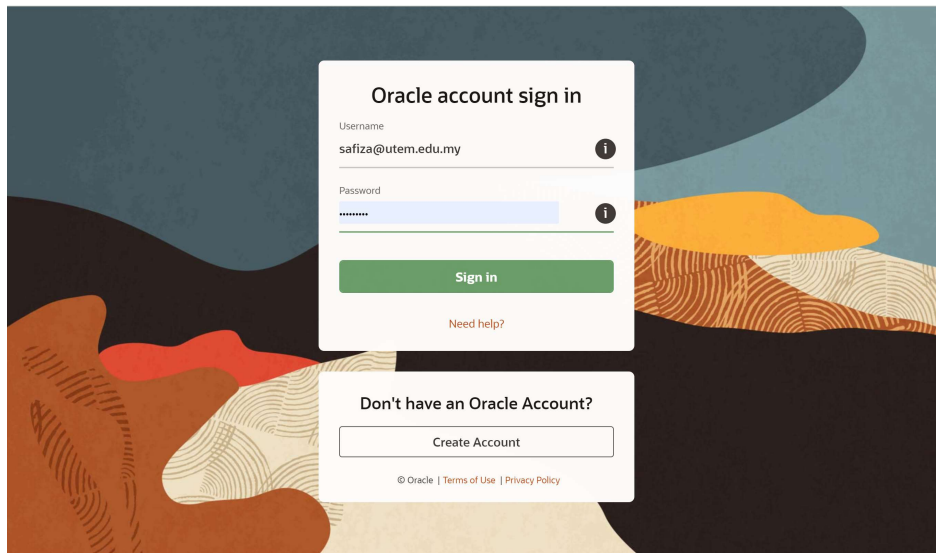
## Practical

### IMPLEMENT SQL USING LIVE SQL

1. Go to the **LiveSQL** website by click [HERE](#).
2. The page below will be displayed.



3. Click the SignIn button at the top-right screen.
4. The page below will be displayed. Insert the related input. If you don't have an Oracle account, you need to create an account first by click Create Account button.



**Oracle account sign in**

Username  
safiza@utem.edu.my

Password  
\*\*\*\*\*

**Sign in**

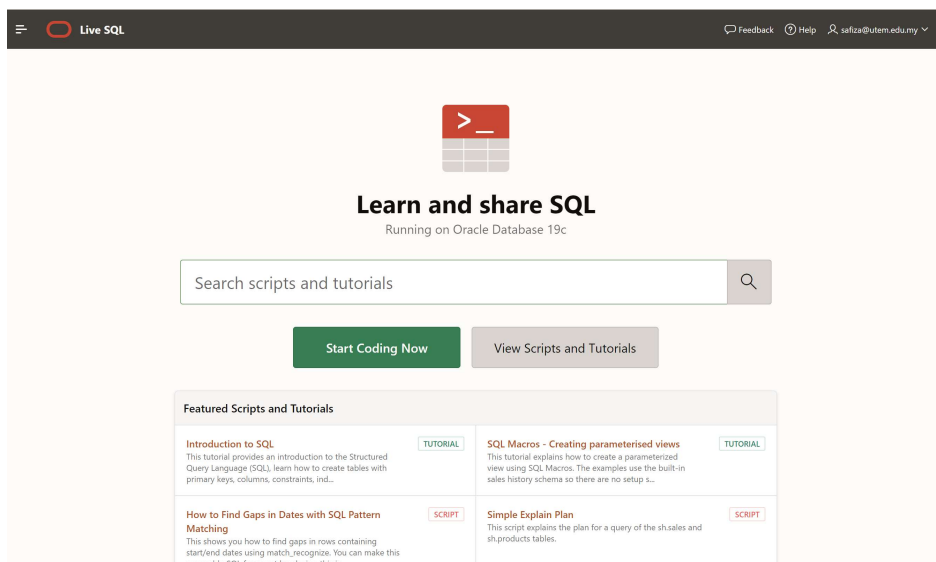
[Need help?](#)

**Don't have an Oracle Account?**

**Create Account**

© Oracle | [Terms of Use](#) | [Privacy Policy](#)

5. Click the Sign-In button. The page below will be displayed.



**Live SQL**

Feedback Help safiza@utem.edu.my

**Learn and share SQL**  
Running on Oracle Database 19c

Search scripts and tutorials

**Start Coding Now** **View Scripts and Tutorials**

**Featured Scripts and Tutorials**

<b>Introduction to SQL</b> This tutorial provides an introduction to the Structured Query Language (SQL), learn how to create tables with primary keys, columns, constraints, ind... <b>TUTORIAL</b>	<b>SQL Macros - Creating parameterised views</b> This tutorial explains how to create a parameterized view using SQL Macros. The examples use the built-in sales history schema so there are no setup s... <b>TUTORIAL</b>
<b>How to Find Gaps in Dates with SQL Pattern Matching</b> This shows you how to find gaps in rows containing start/end dates using match_recognize. You can make this a reusable SQL fragment by placing this in... <b>SCRIPT</b>	<b>Simple Explain Plan</b> This script explains the plan for a query of the sh.sales and sh.products tables. <b>SCRIPT</b>

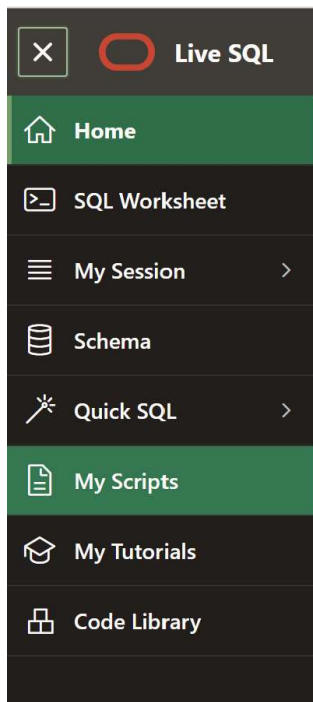
6. Click the hamburger button  at the top left window to expand the navigation options.



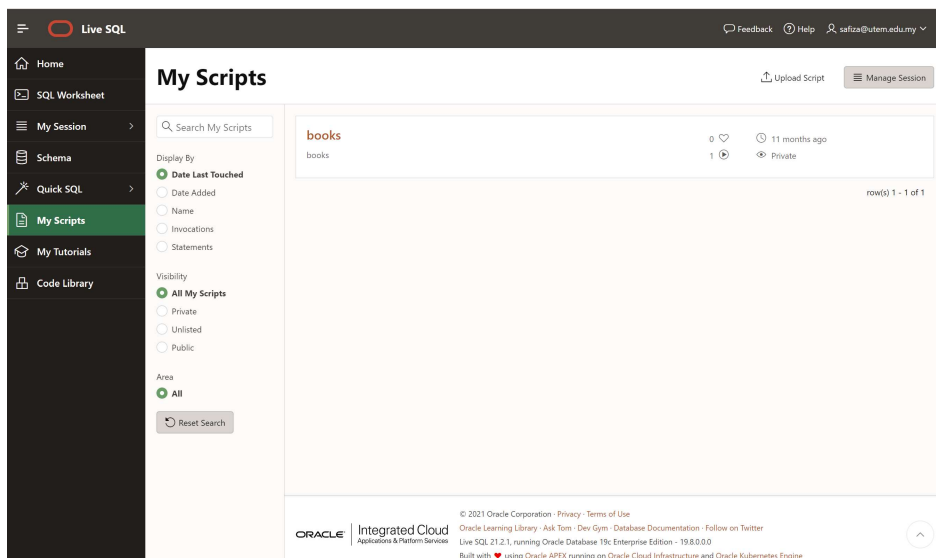
**Live SQL**

Feedback Help safiza@utem.edu.my

7. The list of navigation options will be displayed below. Choose My Script.




8. The below page will be displayed.



9. Click the Upload Script button  Upload Script . Select the Script for Lab Week 9.sql file from your folder. Put the required field (Script Name and Description). Click the Upload Script button.

### Upload Script

×

File \*  


File size is limited to 1 mb.

Script Name \*

Character Set

Description \*

Cancel

 Upload Script

10. Click the Run Script button as shown below.

My Scripts \

## Script

① Actions Edit Attributes Run Script

Name	Week9
Description	Week 9 Practical
Area	-
Visibility	Private - you are the only person who can access
Tags	-
Script Results	We cannot determine the last run date for this script. What's This?
Last Updated	Tuesday May 18, 2021 (Created 42 seconds ago)
Metrics	27 Statements, 2,247 bytes

11. The script results will be displayed as shown below. Ignore the warning that appeared. Click the X button at the top right of the page.

## Script Results

X

Script Week9



### Warning

3 of 27 statements had errors, 24 were successful. 7 objects created.

Statement 1	<pre>DROP TABLE ASSIGNMENT_PROJ CASCADE CONSTRAINTS</pre> <p>ORA-00942: table or view does not exist</p>
Statement 2	<pre>DROP TABLE PROJECT_EMPLOYEE CASCADE CONSTRAINTS</pre> <p>ORA-00942: table or view does not exist</p>
Statement 3	<pre>DROP TABLE PROJECT CASCADE CONSTRAINTS</pre> <p>ORA-00942: table or view does not exist</p>
Statement 4	<pre>CREATE TABLE PROJECT(   ProjectID Number Not Null,   Name Char(25) Unique Not Null,   Department Varchar2(100) Null,   MaxHours Number(6,1) Default 100,   CONSTRAINT ProjectPK Primary Key (ProjectID)</pre>

① Replace Script My Session SQL Worksheet

12. Choose SQL Worksheet from the navigation pane. The page below shows the canvas where you can write your SQL statement.

Live SQL

Feedback Help safira@utem.edu.my

Home SQL Worksheet My Session Schema Quick SQL My Scripts My Tutorials Code Library

SQL Worksheet

Clear Find Actions Save Run

SQL Statement Output

© 2021 Oracle Corporation - Privacy - Terms of Use  
 Oracle Learning Library - Ask Tom - Dev Gym - Database Documentation - Follow on Twitter  
 Live SQL 21.2.1, running Oracle Database 19c Enterprise Edition - 19.0.0.0  
 Built with ♥ using Oracle APEX running on Oracle Cloud Infrastructure and Oracle Kubernetes Engine

13. Write desired SQL statement and click the Run button to get the result.

## SELECT STATEMENT

The SELECT statement is used to retrieve data from one or more objects within a database. For the following example, assume that the sample data has been entered into the database. You can get the DDL script [HERE](#).

### 1. Reading Specified Columns from a Single Table

The following SQL statement will query three (3) of the four (4) columns of the **PROJECT** table.

```
SELECT Name, Department, MaxHours FROM Project;
```

Notice that after the SELECT statement, we may list the name of the columns to be queried with comma-delimited. The result of this statement is shown in Figure 1 below.

NAME	DEPARTMENT	MAXHOURS
Q3 Portfolio Analysis	Finance	75
Q3 Tax Prep	Accounting	145
Q4 Product Plan	Marketing	138
Q4 Portfolio Analysis	Finance	110

4 rows returned in 0.00 seconds [Download](#)

Figure 1: Relation of SELECT Name, Department, MaxHours FROM Project

The result of a SQL SELECT statement is a relation (or table). The order of the column names after the keyword SELECT determines the order of the columns in the resulting table. If you want to change the order of the columns, you can use the previous SELECT statement and change the order as below:

```
SELECT Name, MaxHours, Department FROM Project
```

The result will be:

NAME	MAXHOURS	DEPARTMENT
Q3 Portfolio Analysis	75	Finance
Q3 Tax Prep	145	Accounting
Q4 Product Plan	138	Marketing
Q4 Portfolio Analysis	110	Finance

How about if you want just to obtains only the Department column from the Project table?

Yes, you just write

```
SELECT Department FROM Project;
```

That yield :

DEPARTMENT
Finance
Accounting
Marketing
Finance

4 rows returned in 0.00 seconds [Download](#)

The **Relational Algebra (RA)** for this query is

```
 $\Pi$  Name, Department, MaxHours (Project) --> can be pronounced as Pi Name, Department, MaxHours (Project)
```

Notice that the relation will produce four (4) rows of output. The first and last rows of this table (i.e. Finance) are duplicates. Although the definition of relation has mentioned about duplication is prohibited, however, by default DBMS products do not check for duplication due to time-consuming. Thus, in practice duplicate rows can occur.

In order to eliminate duplicate rows produced by the table, we can use the DISTINCT keywords as shown below:

```
SELECT DISTINCT Department FROM Project;
```

That will yield below relation.

DEPARTMENT
Accounting
Finance
Marketing

3 rows returned in 0.00 seconds [Download](#)

The new relation produces three (3) rows of Department where the duplicate row has been eliminated as desired.

## 2. Reading Specified Rows from a Single Table

SQL statements also can be used to select ALL the columns for certain rows. For example, the SQL statement to obtain ALL of the columns of the Project table for projects sponsored by the finance department is shown below. There are two (2) ways:

### Option 1:

```
SELECT ProjectID, Name, Department, MaxHours FROM Project WHERE Department = 'Finance';
```

The result is:

PROJECTID	NAME	DEPARTMENT	MAXHOURS
1000	Q3 Portfolio Analysis	Finance	75
1500	Q4 Portfolio Analysis	Finance	110

2 rows returned in 0.00 seconds [Download](#)

### Option 2:

A second way to specify all the columns of a table is to use the special character \* (asterisk) after the keyword SELECT. The asterisk (\*) here means ALL COLUMNS.

```
SELECT * FROM Project WHERE Department = 'Finance';
```

The query statement will produce the same result as the previous statement. The result shows a table of all columns of the Project table for rows having a Department value of Finance. The Relational Algebra (RA) for both option 1 and 2 queries above is shown below:

#### RA for Option 1

```
Π ProjectID, Name, Department, MaxHours (σ Department='Finance' (Project))
--> can be pronounced as Pi ProjectID, Name, Department, MaxHours (sigma Department='Finance'(Project))
```

#### RA for Option 2

```
σ Department='Finance' (Project) --> sigma Department='Finance'(Project)
```

## 3. Reading Specifies Columns and Specified Rows from a Single Table

We also can combine the techniques of 1 and 2 from the SELECT statement above to show just some columns and some rows from a table. For example, to obtain only the Name and Department values of employees in the accounting department, we can use this SELECT statement.

```
SELECT Name, Department FROM Project_Employee WHERE Department = 'Accounting';
```

The result is

NAME	DEPARTMENT
Mary Jacobs	Accounting
Rosalie Jackson	Accounting

2 rows returned in 0.02 seconds [Download](#)

The Relational Algebra (RA) for the above query is shown below:

```
Π Name, Department (σ Department='Accounting' (Project_Employee))
```

The pattern SELECT/ FROM/ WHERE is the fundamental pattern of SQL SELECT statements. Many different conditions can be placed in a WHERE clause. for example, the query :

```
SELECT * FROM Project WHERE MaxHours > 100;
```

It will produce a table that displays all columns from the Project table where the MaxHours column is greater than 100 as shown below:

PROJECTID	NAME	DEPARTMENT	MAXHOURS
1200	Q3 Tax Prep	Accounting	145
1400	Q4 Product Plan	Marketing	138
1500	Q4 Portfolio Analysis	Finance	110

3 rows returned in 0.01 seconds [Download](#)

### What is RA for the above statement?

Another use of the WHERE clause is to specify that a column should have one of a set of values. For example, to retrieve the name, phone, and department of employees in either department Accounting or Marketing or Finance, we can use the query as below. There are two options:

#### Option 1: Using OR

```
SELECT Name, Phone, Department FROM Project_Employee WHERE Department = 'Accounting' OR Department = 'Finance' OR Department = 'Marketing';
```

#### Option 2: Use the IN keywords

```
SELECT Name, Phone, Department
FROM Project_Employee
WHERE Department IN ('Accounting', 'Finance', 'Marketing');
```

Both options above yield the same output as below:

NAME	PHONE	DEPARTMENT
Mary Jacobs	285-8879	Accounting
Keni Numoto	287-0098	Marketing
Heather Jones	287-9981	Finance
Rosalie Jackson	285-1273	Accounting
Kim Sung	287-3222	Marketing

5 rows returned in 0.01 seconds [Download](#)

This table shows a relation (table) that has a Department value equal to Accounting, Finance, or Marketing.

In order to select rows that do not have department value with any of these, we can use NOT IN as follows:

```
SELECT Name, Phone, Department
FROM Project_Employee
WHERE Department NOT IN ('Accounting', 'Finance', 'Marketing');
```

The result of the above query is shown below:

NAME	PHONE	DEPARTMENT
James Nestor	null	Info System
Richard Wu	287-0123	Info System

2 rows returned in 0.02 seconds [Download](#)

Notice that the difference between IN and NOT IN is, when using IN, the column may equal ANY of the values in the list (e.g. Accounting, Finance, Marketing). However, when using NOT IN, the column must not be equal to ALL of the values in the list (i.e. exclude Accounting, Finance, and Marketing from the query).



## RANGES, WILDCARDS, and NULLS in WHERE CLAUSES

WHERE clauses can refer to ranges and partial values.

### 1) BETWEEN

The keyword BETWEEN.. AND is used for ranges. For example, the statement:

```
SELECT Name, Department FROM Project_Employee WHERE EmployeeNumber BETWEEN 200 AND 500
```

will produce the following result:

NAME	DEPARTMENT
Keni Numoto	Marketing
Heather Jones	Finance
Rosalie Jackson	Accounting
James Nestor	Info System

4 rows returned in 0.00 seconds [Download](#)

This statement is equivalent to:

```
SELECT Name, Department FROM Project_Employee WHERE EmployeeNumber => 200 AND EmployeeNumber <= 500
```

Thus, the end values of BETWEEN (in this case 200 and 500) are included in the selected range.

### 2) LIKE (Pattern Matching)

The keyword LIKE is used in SQL expressions to select partial values. The underscore symbol (\_) represents a single, unspecified character. Oracle handles special characters with the ESCAPE clause, and the most common ESCAPE is for the wildcard percent sign (%), and the underscore (\_). It can be used to find values that fit a pattern as in the following.

```
SELECT * FROM PROJECT WHERE Name LIKE 'Q_ P%'
```

For example, by using underscores and %, this query wants to display all values that contain the string "Q\_ P" (e.g. "Q<<any value>>P" within a query). The underscore means that any character can occur in the spot. The result of this statement is:

PROJECTID	NAME	DEPARTMENT	MAXHOURS
1000	Q3 Portfolio Analysis	Finance	75
1400	Q4 Product Plan	Marketing	138
1500	Q4 Portfolio Analysis	Finance	110

3 rows returned in 0.01 seconds [Download](#)

This result fulfills the values required in which the project Name starts from alphabet Q with followed by any number character, followed by space, and alphabet P.

#### How to find all the project name which has word Portfolio on it?

Answer in Forum under topic SQL QUERY - Week 9.

To find all employees who have a Phone value that begins with '285-', we can use four (4) underscores to represent any last four digits as follows:

```
SELECT * FROM Project_Employee WHERE Phone LIKE '285-____';
```

The result is :

EMPLOYEEENUMBER	NAME	PHONE	DEPARTMENT
100	Mary Jacobs	285-8879	Accounting
400	Rosalie Jackson	285-1273	Accounting

2 rows returned in 0.01 seconds [Download](#)

This statement is equivalent to:

```
SELECT * FROM Project_Employee WHERE Phone LIKE '285-%';
```

The percent sign (%) is used to represent a series of one or more unspecified characters. In this example statement, the % sign is replacing four underscores in the previous statement.

The result is the same as in the previous example.

If we want to find all the employees who work in a department that end in *ing*, we would use the % character as follows:

```
SELECT * FROM Project_Employee WHERE Department LIKE '%ing';
```

The result is:

EMPLOYEENUMBER	NAME	PHONE	DEPARTMENT
100	Mary Jacobs	285-8879	Accounting
200	Keni Numoto	287-0098	Marketing
400	Rosalie Jackson	285-1273	Accounting
700	Kim Sung	287-3222	Marketing

4 rows returned in 0.01 seconds [Download](#)

### 3) IS NULL

The keyword *IS NULL* can be used in the WHERE clause to search for null values. The following SQL will find the names and departments of all employees having a null value for Phone (employees who have no phone number).

```
SELECT Name, Department FROM Project_Employee WHERE Phone IS NULL;
```

The result of this query is :

NAME	DEPARTMENT
James Nestor	Info System

1 rows returned in 0.01 seconds [Download](#)

### SORTING THE RESULT

The order of rows in the result of a SELECT statement is arbitrary. If this is undesirable, the ORDER BY phrase can be used to sort the rows. For example, the following will display the names and departments of all employees sorted by the Department.

```
SELECT Name, Department FROM Project_Employee ORDER BY Department
```

The result is:

By default, SQL will sort in ascending order. The keywords *ASC* and *DESC* can be used to specify ascending and descending order when necessary. Thus to sort employees in descending order by Department, use this SQL:

```
SELECT Name, Department FROM Project_Employee ORDER BY Department Desc;
```

The result is:

NAME	DEPARTMENT
Keni Numoto	Marketing
Kim Sung	Marketing
Richard Wu	Info System
James Nestor	Info System
Heather Jones	Finance
Mary Jacobs	Accounting
Rosalie Jackson	Accounting

7 rows returned in 0.01 seconds [Download](#)

If there are two or more columns that need to be sorted, for example, to sort the employee name and department first in descending order by Department value, and then within Department, we want it sort in ascending order by Name, we would specify as follows:

```
SELECT Name, Department FROM Project_Employee ORDER BY Department Desc, Name Asc;
```

The result is:

NAME	DEPARTMENT
Keni Numoto	Marketing
Kim Sung	Marketing
Richard Wu	Info System
James Nestor	Info System
Heather Jones	Finance
Mary Jacobs	Accounting
Rosalie Jackson	Accounting

7 rows returned in 0.01 seconds

[Download](#)