实验1　MYSQL数据库SQL语句练习实验

实验类别：验证性　　实验级别：必做

开课单位：计算机与软件学院　　实验时数：12学时

一、实验目的：

1、了解DBMS　系统的功能、软件组成；

2、掌握利用SQL　语句定义、操纵数据库的方法。

二、实验要求：

1、在课外安装相关软件并浏览软件自带的帮助文件和功能菜单，了解DBMS　的功能、结构；

2、创建一个有两个关系表的数据库；(建议采用MYSQL)

3、数据库、关系表定义；

4、学习定义关系表的约束(主键、外键、自定义)；

5、了 解SQL 的数据定义功能；

6、了解SQL 的操纵功能；

7、 掌握典型的SQL 语句的功能；

8、 了解视图的概念；

三、实验设备：

计算机、数据库管理系统如MYSQL等软件。

四、建议的实验步骤：

0、安装MYSQL 软件。见附件1

1、使用SQL　语句建立关系数据库模式及数据如下；(注：**数据要自己输入**)

EMP:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-Dec-90	13750		20
7499	ALLEN	SALESMAN	7698	20-FEB-89	19000	6400	30
7521	WARD	SALESMAN	7698	22-FEB-93	18500	4250	30
7566	JONES	MANAGER	7839	02-APR-89	26850		20
7654	MARTIN	SALESMAN	7698	28-SEP-97	15675	3500	30
7698	BLAKE	MANAGER	7839	01-MAY-90	24000		30
7782	CLARK	MANAGER	7839	09-JUN-88	27500		10
7788	SCOTT	ANALYST	7566	19-APR-87	19500		20
7839	KING	PRESIDENT		17-NOV-83	82500		10
7844	TURNER	SALESMAN	7698	08-SEP-92	18500	6250	30
7876	ADAMS	CLERK	7788	23-MAY-96	11900		20
7900	JAMES	CLERK	7698	03-DEC-95	12500		30
7902	FORD	ANALYST	7566	03-DEC-91	21500		20
7934	MILLER	CLERK	7782	23-JAN-95	13250		10
3258	GREEN	SALESMAN	4422	24-Jul-95	18500	2750	50
4422	STEVENS	MANAGER	7839	14-Jan-94	24750		50
6548	BARNES	CLERK	4422	16-Jan-95	11950		50

DEPT:

DEPTNO	DNAME	LOC
10	ACCOUNTING	LONDON
20	RESEARCH	PRESTON
30	SALES	LIVERPOOL
40	OPERATIONS	STAFFORD
50	MARKETING	LUTON

2、用 SQL 定义数据库的关系表；

注：（每位同学在各自创建的图表名字后面添加自己学号以示区分，如EMP20170000112 等 ）

3、定义各个关系的字段和自定义的数据完整性约束；

4、确定关系表的主键、外键；

5、对照帮助文件和教材理解主键和外键的约束规则；

6、分别为关系表添加记录；

7、理 解SQL 语句和关系运算的关系；

8 、练习典型的 SQL 语句，对第6步实验中已建立的表做查询、插入、更新、删除等操作；完成练习题。

9、检查同学作业的思考题是否存在问题。

注：以上具体步骤可参见帮助文件SQL handbook 或相关书籍。

附件一、WampServer 安装

1. 安装软件

建议使用WampServer, 这个集成环境对内含的组件的配置做了比较大的优化。

下载地址：

http://www.wampserver.com/en/

WampServer 中的 phpmyadmin 默认是以无密码方式访问MySQL 的，如果要更改为以密码访问方式，步骤如下：

a 启 动WampServer

b 通过在系统托盘中WampServer 的图标上使用鼠标菜单进入phpmyadmin, 点击“权限”按钮，添加新用户并为其设置密码(建议也为root 添加密码，或者干脆只为root 添加密码)一一不建议在平常操作中使用root, 太暴力了 ……

c 关掉网页，关掉WampServer

d 进入<wamp 安装目录>/apps/phpmyadmin3.4.10.1, 修改 config.inc.php

-->把$cfg['Servers'][Si]['auth_type']='config'; 改为$cfg['Servers'][$i][auth_type']='cookie';

-->把$cfg['Servers'][Si]['AllowNoPassword']=true; 改为$cfg['Servers'][$i]['AllowNoPassword']=false;

—>在<?php ?>中的任意位置添加$cfg['blowfish_secret']='<任意字符串>;

5 经过以上四步即可把从 phpmyadmin 登 录 MySQL 的方式改为密码验证，重新启动 WampServer 即可

可能遇到的问题如下：

a.修改phpmyadmin 的配置之后出现无法打开phpmyadmin 页面

-->删除浏览器历史记录(特别是IE8及以下的版本。 一般建议，既然用web 的话，那就换其它浏览器吧，FireFox,chrome,opera 等都很好)

-->删除<wamp 安装目录>/tmp 下的所有文件，重新启动WampServer

b、从一开始就无法打开phpmyadmin 页面，提示permission denied 之类的信息

-->win7 下可能会有此问题，解决方法是修改<wamp 安装目录>/alias/phpmyadmin.conf, 把

Deny from all

Allow from 127.0.0.1

改为

Allow from all

附件二、SQLhandbook

CONTENTS

This book has been designed to help you learn SQL as it has to be learnt by doing,not by teaching.

It is therefore in your best interest to work your way through it (10-15 hours work)systematically.

You may be asked to submit some of the answers to the exercises,for an assignment and may be asked specific details in an exam.

SQL (Structured Query Language)is a relational database language.Amongst other things the language consists of statements to insert,update,query and protect data. Although SQL is not a DBMS,for simplicity in this manual SQL will be considered as a DBMS as well as a language.Of course,in the places where it is necessary,a distinction will be drawn.

There are a few things to note about SQL as a database language,because it is a relational database language,SQL may be grouped with the non-procedural database languages.By non-procedural it is meant that users(with the help of the various statements)have only to specify which data they want and not how this data must be found.C++,Java and VB are examples of Procedural languages.It also means that there are no variables,IF statements or loop constructs.Because it is non procedural,it is very difficult to teach,and the only way to learn it is by working through this book and picking up how certain results can be achieved.

SQL can be used in two ways.First,interpretively:an SQL statement is entered at a terminal or PC and immediately processed or interpreted.The result is also visible immediately.This is known as interactive SQL.The second way is known as embedded SQL.The SQL statements are embedded in a program written in another, procedural language.Results of these statements are not immediately visible to the user,but are processed by the 'enveloping'program.In this module we shall be

assuming the interpretive use of SQL.

SQL has already been implemented by many manufacturers as the database language for their DBMS.It is not the case,therefore,that SQL is the name of a particular manufacturer's product available on the market today.However,it is the market standard and you will find many career opportunities within the general SQL field. Currently it is number one in the Jobs market.

Some manufacturers are now providing SQL-server machines.These machines can be connected to a DBMS,and they carry out all the database functions defined in SQL. Thus SQL is now a data interchange language between any systems that can 'speak SQL.Typically,an SQL-server is placed on a LAN where it processes all database operations for clients on the LAN.

Please note that,although SQL is an ISO standard,each manufacturer have their own add-ons.

SELECTING DATA FROM TABLES.

The SELECT Command is the basis of all queries on tables,therefore its full description is given to show its power.Examples of the various formats are provided after the description.

```
SELECT
  column_1,column_2,..
FROM
  table_1
[INNER|LEFT |RIGHT]JOIN table_2 ON conditions
WHERE
  conditions
GROUP BY column_1
HAVING group_conditions
ORDER BY column_1
LIMIT offset,length;
```

The SELECT statement consists of several clauses as explained in the following list:

- SELECT followed by a list of comma-separated columns or an asterisk(*)to indicate that you want to return all columns.
- FROM specifies the table or view where you want to query the data.
- JOIN gets related data from other tables based on specific join conditions.
- WHERE clause filters row in the result set.
- GROUP BY clause groups a set of rows into groups and applies **aggregate functions** on each group.
- HAVING clause filters group based on groups defined by GROUP BY clause.
- ORDER BYclause specifies a list of columns for sorting.
- LIMIT constrains the number of returned rows.

TheSELECT and FROM clauses are required in the statement.

Description: Selects rows and columns from one or more tables.May be used as a command,or(with certain restrictions on Clauses)as a subquery in another **SELECT,** and **UPDATE,** or other **SQL** command.

Don't worry too much about this generic syntax list as you will see all kinds of examples throughout this book.

PARAMETERSAND CLAUSES.

ALL makes **SELECT** display all rows produced by the query.Since this is the default,it is generally not needed

DISTINCT makes it omit duplicate rows.

* makes **SELECT** display all columns of the table(s)specified by **FROM,**in the order they were defined when the table(s)were created.

i.e.SELECT*FROM EMP;

Alternatively,each expression becomes one column in the display.

i.e.SELECT EMPNO,ENAME FROM EMP;

displays only the named columns in the expression.

Each alias,if specified,is used to label the preceding expression in the displayed table.

e.g.SELECT ENAME“Name”,SAL“Salary”from EMP;

Note the use of double quotes here.character strings are delimited by single quotes.

FROM table specifies the table or view to be drawn on.More than one table implies a join. **Alias,** if specified,may be used as an alias for the preceding table through the rest of the **SELECT** command.

SELECTED EXAMPLES AND WORKSHEETS.

The examples in this book should be worked through carefully to ensure that you understand what the commands are doing. Your assignment work will assume that knowledge.

You will need the following tables,EMP and DEPT.

These can be created with the following commands:

EPT;

The data in them is currently as shown on the next page:

Note -if the table contents become corrupted (particularly after the Update example in Exercise 7),you can always delete the tables and start again.This can be achieved by:

```
DROP TABLE EMP;
DROP TABLE DEPT;
```

To list all the tables in your Oracle area:

```
SHOW TABLES;
```

THE DATA USED IN THESE EXERCISES:

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7369 SMITH CLERK 7902 17-DEC-9013750 20
7499 **ALLEN SALESMAN 7698 20-FEB-89 19000**
6400 30
7521 WARD SALESMAN 7698 22-FEB-93 18500
4250 30
7566 JONES MANAGER 7839 02-APR-89
26850 20
7654 **MARTIN SALESMAN 7698 28-SEP-97**
15675 3500 30
7698 BLAKE MANAGER 7839 01-MAY-90 24000
30
7782 **CLARK MANAGER 7839 09-JUN-88 27500**
10
7788 SCOTT ANALYST 7566 19-APR-87 19500
20
7839 KING PRESIDENT 17-NOV-83
82500 10
7844 TURNER SALESMAN 7698 08-SEP-92 18500
6250 30
7876 ADAMS CLERK 77 88 23-MAY-96
11900 20
7900 JAMES CLERK 7698 03-DEC-95
12500 30
7902 FORD ANALYST 7566 03-DEC-91
21500 20
7934 MILLER CLERK 7782 23-JAN-95
13250 10
3258 GREEN SALESMAN 4422 24-JUL-95 18500 2750 50
4422 STEVENS MANAGER 7839 14-JAN-94 24750 50
6548 BARNES CLERK 4422 16-JAN-95 11950 50

DEPTNO DNAME LOC

10 ACCOUNTING LONDON
20 RESEARCH PRESTON
30 SALES LIVERPO0L
40 OPERATIONS STAFFORD
50 MARKETING LUTON

THE SELECT STATEMENT

The SELECT statement is the workhorse of query processing the basic statement is:-

SELECT COLUMN(S)**FROM** TABLENAME;

This is the minimum amount of detail which must be entered for a **SELECT** statement to work.

Try the following examples:-

SELECT*FROM emp;

Provides a listing of all the data (all columns)in the EMP table.

SELECT ename FROM emp;

Gives a list of all the employee names found in the emp table.

SELECT dname,loc FROM dept;

gives department names and locations.

SELECT job FROM emp; (with duplicates)

Lists all the jobs in the emp table even if they appear more than once.

SELECT DISTINCT job FROM emp; (without duplicates)

List all the jobs in the EMP table eliminating duplicates.

SELECT job,deptno FROM emp; (with duplicates)

Lists the combination ofjobs and departments for every row of the emp table.

SELECT DISTINCT job,deptno FROM emp; (without duplicates)

List all the combinations of job and department in the EMP table eliminating duplicates.

THE WHERE CLAUSE

A WHERE clause causes a 'search'to be made and only those rows that meet the search condition are retrieved.

A WHERE clause condition can use any ofthe following comparison operators:-

= equal to

```
SELECT*FROM emp
     WHERE ename='JONES';
```

(Again note that the data is case sensitive,this would not find Jones)

!=not equal to

^=not equal to

<>not equal to

```
SELECT*FROM emp
     WHERE ename !='FORD';
```

> greater than

```
SELECT*FROM emp
     WHERE sal>15000;
```

>=greater than or equal to

```
SELECT*FROM   emp
     WHERE sal>=12000;
```

< less than

```
SELECT*FROM emp
     WHERE sal<15000;
```

<=less than or equal to

```
SELECT*FROM emp
     WHERE sal<=12000;
```

or special SQLoperators

BETWEEN lowAND high (values are inclusive)

SELECT*FROM emp
WHERE sal BETWEEN 10000 AND 15000;

IN(VALUE1,VALUE2,VALUE3....) **c**haracter strings must be enclosed in quotes

SELECT*FROM emp
WHERE job IN('CLERK','ANALYST);

Selects all employees who are

Clerks or analysts

LIKE 'string picture' use '%'and _'as wildcards within a string picture.Each_acts for one character.

SELECT*FROM emp
WHERE ename LIKE'%A%';%is for any number of characters

Selects all employees with an'A'

in their name.

IS NULL IS may only be used with NULL's (this means the variable has no value)

and also **NOT** any of the above expressions (used for negation purposes).

Try the following:-

SELECT ename,empno,deptno
FROM emp
WHERE job='CLERK';

List the names,numbers and departments of all the Clerks.

SELECT ename,sal,comm FROM emp
WHERE comm>sal;

Find the employees whose commission is greater than their salary.

SELECT ename,job,sal FROM emp
WHERE sal BETWEEN 12000 AND 14000;

Finds all employees who earn between 12,000 and 14,000

Selecting rows within a range,the WHERE clause can have a low-value and a high-value associated with it,these values represent the bottom and top of the required range.

NOT BETWEEN means that only rows that are outside the range will be selected.

SELECT ename FROM emp
WHERE job IN('CLERK','ANALYST','SALESMAN);

Finds the employees who are clerks,analysts or salesmen.

NOT IN would list those employees whose jobs are not in the list.

SELECT ename FROM emp

WHERE job NOT IN('CLERK','ANALYST','SALESMAN');

SELECT ename,deptno FROM emp
WHERE ename='FORD';

Finds the departments that employees called Fordwork in.

SELECT ename,deptno FROM emp
WHERE ename LIKE'__A%';

Finds employee names that have an A as the 3rd letter i.e.Blake,Clark etc. (Note -there are 2 underscores before the A)

SELECT ename FROM emp
WHERE comm IS NULL;

Finds all employees that do not have any commission

Multiple search conditions may be used in a select statement,linked by either **AND** (both statements must be true for a row to be selected)or **OR** (only one condition

must be true for a row to be selected) AND and OR may be combined to produce complex search conditions and for clarity and reliability should be parenthesised to force precedence.Otherwise normal computing rules apply.

```
SELECT*FROM   emp
    WHERE  job='MANAGER'
    OR   job='CLERK'
    AND  deptno=10;
```

Find everyone whose job title is manager,and all the clerks in department 10

```
SELECT*FROM   emp
    WHEREjob='MANAGER'
    OR(job='CLERK'
    AND  deptno=10);          (use of parentheses to clarify.)
```

```
SELECT*FROM   emp
    WHERE   (job='MANAGER'
    OR   job='CLERK')
    AND  deptno=10;
```

Find all the managers or clerks in department 10.

Any group of search conditions can be negated by enclosing the statement in parentheses and preceding them with NOT.

```
SELECT*FROM   emp
    WHERE NOT (job='MANAGER'
    ORjob='CLERK')
    AND  deptno=10;
```

Find anyone who is neither a manager nor a clerk but is in department 10.

THE ORDER BY CLAUSE

By default Oracle will display rows of data in a totally unordered way.The ORDER BY clause should be used to impose an ordering of the rows retrieved by a query and should always be placed last in the query(or query block).

The use of **ORDER BY** causes data to be sorted (by default)as follows:-

NUMERICS ascending order by value

DATES chronological order

CHAR alphabetically

The keyword **DESC** causes the sort to be reversed.

NULL values in a sorted column will always be sorted high,i.e.they will be first when values are sorted in descending order and last when sorted in ascending order.

```
SELECTempno,ename,hiredate FROM emp
    ORDER BY hiredate;
```

Shows details of employees with earliest hiredates first.

```
SELECT job,sal,ename FROM emp
    ORDER BYjob,sal DESC;
```

To order all employees by job,and within job,put them in descending salary order;

```
SELECT ename,job,sal,comm,deptno FROM emp
    ORDER BY3;
```

Lists employees in salary order (salary is the 3rd item in the SELECT list)

EXERCISES.1 SIMPLE COMMANDS

1 List all information about the employees.

2 List all information about the departments

3 List only the following information from the EMP table(Employee name,employee number,salary,department number)

4 List details of employees in departments 10 and 30.

5 List all the jobs in the EMP table eliminating duplicates.

6.What are the names of the employees who earn less than f20,000?

7.What is the name,job title and employee number ofthe person in department 20 who earns more than f25000?

8.Find all employees whose job is either Clerk or Salesman.

9.Find any Clerk who is not in department 10.

10.Find everyone whose job is Salesman and all the Analysts in department 20.

11.Find all the employees who earn between f15,000 and f20,000.
Show the employee name,department and salary.

12 Find the name of the President.

13 Find all the employees whose last names end with S

14 List the employees whose names have TH or LL in them

15 List only those employees who receive commission.

16 Find the name,job,salary,hiredate,and department number of all employees by alphabetical order of name.

17.Find the name,job,salary,hiredate and department number of all employees in ascending order by their salaries.

18.List all salesmen in descending order by commission divided by their

salary.

19.Order employees in department 30 who receive commision,in ascending order by commission

20 Find the names,jobs,salaries and commissions ofall employees who do not have managers.

21 Find all the salesmen in department 30 who have a salary greater than or equal to f18000.

JOINING TABLES

It is necessary to join two or more tables for some queries.This takes place by establishing a relationship (usually equality)between a column(domain)present in two tables known as a foreign key.Simple joins are usually called **equi-joins.** A join is automatically performed when a reference is made to more than one table in the **FROM** clause

```
SELECT ename,sal,loc FROM emp,dept
    WHERE ename='ALLEN'       (search condition)
    AND emp.deptno=dept.deptno;        ((join condition)
```

Find Allen's name and salary from the EMP table and location of Allen's department from the DEPT table.

N.B.because we are now referencing two tables which each have a column with the same name (deptno),we must always qualify deptno with its table name in order to prevent confusion,this qualification must be used whenever ambiguous column names are used within an SQL statement.

```
SELECT ename,dname FROM emp,dept
    WHERE emp.deptno =dept.deptno
    ORDER BY ename;
```

List the name and department of all employees in name order.(Thisjoins the two tables over DEPTNO and projects out ENAME and DNAME)

Abbreviating Table Names.

Table names can be abbreviated in order to simplify what is typed in with the query. In this example E and D are abbreviated names for emp and dept.

List the department name and all employee data for employees that work in Chicago;

```
SELECT dname,E.*FROM emp E,dept D
    WHERE E.deptno=D.deptno AND loc='LUTON'
        ORDER BY E.deptno;
```

Note-if we didn't have ORDER BY E.deptno,but had ORDER BY deptno

We would get a syntax error because it would know whether to sort on the deptno in Emp or Dept.

Joining a Table to Itself

A table label can be used for more than just abbreviating a table name in a query.It also allows a join of a table to itself as though it were two separate tables.This can be very useful because a single SELECT will only go through a table once.By having two copies of the same table,you can find a specific record in the first copy and then search the second copy for comparisons.

```
SELECT WORKER.ename,WORKER.sal
    FROM emp WORKER,emp MANAGER
    WHERE WORKER.mgr=MANAGER.empno
    AND WORKER.sal>MANAGER.sal;
```

In the query the **emp** table is treated as if it were two separate tables named **WORKER** and MANAGER.

First all the **WORKERS** are joined to their **MANAGERS** using the **WORKER's** manager's employee number(**WORKER.mgr)** and the **MANAGER's** employee number**(MANAGER.empno).**

The **WHERE** clause eliminates all **WORKER MANAGER** pairs except those where the **WORKER** earns more than the manager **(WORKER.SAL >MANAGER.SAL).**

Find all employees that earn more than Jones.

```
SELECT  X.ename,X.sal,X.job,Yjob,Y.ename,Y.sal
    FROM emp X,emp Y
        WHERE  X.sal>Y.sal
        AND  Yename='JONES';
```

i.e.find JONES,and then go through the table again comparing.

Selecting all possible combinations of rowS.

If the **WHERE** clause contains no join condition,then all possible combinations of rows from tables listed in the from clause are displayed.The result (Cartesian product) is normally not desired so a join condition is usually specified.

This is a common error,and to be avoided because if table A has 20 rows and table B has 30 rows then not using a join would result in 600 output lines.

Join the Allen row from the EMP table with all the rows in the Dept table

```
SELECT ename,loc FROM emp,dept
    WHERE ename ='ALLEN';
```

OUTER JOINS

When processing joins between emp and dept you will notice that details of department 40 never appear in the output.This is because department 40 has no corresponding rows in the emp table and therefore cannot take part in the join.If it is required to include records which are outside of the relationship between tables an **outer join** must be used.

SELECT dept.deptno,dname,ename,sal from dept left outer join emp on dept.deptno=emp.deptno

The l**eft【outer】 join** effectively adds a dummy row to the emp table for each department record which has no corresponding employees.The department record is then joined with this dummy row and appears once in the output,having nulls in any columns from the emp table.

EXERCISES 2 JOINS

1.Find the name and salary of employees in Luton.

2.Join the DEPT table to the EMP table and show in department number order.

3.List the names of all salesmen who work in SALES

4.List all departments that do not have any employees.

5 For each employee whose salary exceeds his manager's salary,list the employee's name and salary and the manager's name and salary.

6.List the employees who have BLAKE as their manager.

SQL FUNCTIONS

SQL*PLUS has a wide range of functions which may be applied to Oracle data.There are four classes of functions:-

string functions for searching and manipulating strings.
arithmetic functions for performing calculations on numeric values
date functions for reformatting and performing data arithmetic
aggregate functions for calculations on groups ofdata.

Useful string functions

NOTE -when you wish to Select something,but the data is not in a table (as the examples below),you can use a dummy table name called DUAL.This table is only recognised by Mysql as a dummy table,and will never appear as an actual structure.MySQL may ignore the clauses.MySQL does not require FROM DUALif no tables are referenced.

LOWER(string) converts upper case alphabetic characters to lower case.Other characters are not affected

SELECT LOWER('MR.SAMUEL HILLHOUSE')
FROM DUAL:
gives mr samuel hillhouse

UPPER(string) converts lowercase letters in a string to uppercase.

SELECT UPPER('Mr.Rodgers')FROM DUAL;

SUBSTR(string,startposition,length)shows a part of the string starting at the start position of the specified length

SELECT SUBSTR('ABCDEF',2,3)FROM dual;
gives BCD

INSTR(string1,string2) finds the start position of one string inside another string
SELECT INSTR('ABCDEF','DEF)FROM dual;

gives 4

str_to_date (string[,format]) converts the string to a date.A format may optionally be specified (see later)

LPAD(str,len,padstr)left pads the string with the specified fill characters to the specified length.

SELECT LPAD('hi',4,'??');

gives '??hi'

RPAD(str, len, padstr) right pads the string with the specified fill characters to the specified length.

LTRIM(string) Returns the string str with leading space characters removed

SELECT LTRIM(barbar');
WOULD GIVE barbar

RTRIM(string) Returns the string str with trailing space characters removed

TRIM([{BOTH|LEADING|TRAILING}[remstr]FROM]str)
Returns a string after removing all prefixes or suffixes from the given string.

MySQL TRIMO function

Syntax:

TRIM([[{LEADING|TRAILING|BOTH}]
[remstr]FROM]str

Example: trailing M will be remove

TRIM(TRAILING M FROM 'MADAM)

Result:MADA

Example: leading M will be remove

TRIM (L EADING M FROM MADAM)

Result:ADAM

Example: both leading and trailing M will be remove

TRIM (BOTH M FROM MADAM)

Result:ADA

Example: both leading and trailing M will be remove

position not specified

TRIM (M FROM MADAM)

Result:ADA

Example: both leading and trailing white spaces will be remove

Result:MADAM

IFNULL(expression1,expression2); takes two expressions and if the first expression is not NULL,it returns the first expression.Otherwise,it returns the second expression.

e.g.SELECT*,IFNULL(Comm,0)FROM EMP;

LENGTH(char) length in characters of specified string

Remember --you must put single quotes round all data items which are strings.

EXAMPLES OF STRING FUNCTIONS

SELECT SUBSTR(ename,1,4)FROM emp;

UPPER(dname)

SELECT UPPER('helen campbell')FROM dual;

LOWER(ename)

SELECT LOWER('Mr Donald Briffet')FROM dual;

STR_TO_DATE('12-12-92','%d-%m-%Y')

SELECT STR_TO_DATE('12-06-1996','%d-%m-%Y')FROM dual;

LPAD(ename,10,'')

SELECT LPAD(ename,10,"')FROM emp;
(pads the name out to 10 chars with spaces before)

RPAD(ename,10,"')

SELECT RPAD(ename,10,"')FROM emp;
(as above but with spaces after)

LTRIM(ename,'')

SELECT LTRIM(ename,'')FROM emp;
(removes spaces from before the name)

RTRIM(ename,'')

SELECT RTRIM(ename,'')FROM emp;

IFNULL(comm,0)

SELECT IFNULL(comm,0)FROM emp;

(if an employee has no commission then 0 is displayed)

LENGTH(ename)

SELECT LENGTH('Anderson')FROM dual;

NOTE You can also rename columns within the SQL statement

SELECT ename Employee FROM emp;

w**ill output the present ename values with the heading Employee.Note that if the new name is a single word then double quotes are not needed.**

ARITHMETIC FUNCTIONS

ABS(numeric) absolute value of the number

SELECT ABS(-15)"Absolute"FROM DUAL;

MOD(num1,num2)returns the remainder when numl is divided by num2

SELECT MOD(7,5)"modulo"FROM DUAL;

ROUND(numeric[,d]) rounds the number to d decimal places,the rounding can occur to either side of the decimal point.

SELECT ROUND(15.193,1)"round"FROM DUAL;

TRUNCATE(numeric[,d]))truncates to d decimal places,

SELECT TRUNCATE(15.79,1)"truncate"FROM DUAL;

CEIL(numeric)rounds the number up to the nearest integer

SELECT CEIL(10.6)FROM dual;

FLO0R(numeric) truncates the number to the nearest integer

SELECT FLOOR(10.6)FROM dual;

SQRT(numeric)returns the square root of the number (returns NULL if the number is negative)

SELECT SQRT(25)FROM dual;

TO_CHAR(numeric[,format])converts a number to a character string in the specified format

SELECT TO_CHAR(sysdate(),'DY')FROM dual;

SELECT TO_CHAR(sysdate(),'MONTH')FROM dual;

DATE_FORMAT(date,format) **Cut and Paste date_format strings for MySQL**

SELECT DATE_FORMAT(NOw(,'%Y-%m-%d %H:%i:%s')

SELECT DATE_FORMAT(NOw0,'%m')

Some more examples

SIGN(sal -comm)

SELECT SIGN(sal-4*comm)FROM emp;

ABS(sal-comm)

SELECT ABS(sal-4*comm)FROM emp;

ROUND(sal,2)

SELECT ROUND(1234.5678,2)FROM dual;

TRUNCATE(comm,3)

SELECT TRUNCATE(comm,3)FROM emp;

GREATEST(sal,comm)

SELECT GREATEST(sal,comm)FROM emp;

DATE_FORMAT(date,format) Cut and Paste date_format strings for MySQL

SELECT DATE_FORMAT(NOw(,'%Y-%m-%d %H:%i:%s')

IT IS VERY IMPORTANT TO NOTE THAT IF ANY VARIABLE CONTAINS A NULL VALUE THEN ANY SQL STATEMENT INVOLVING ARITHMETIC WILLIGNORE IT

E.G.

SELECTABS(SAL-COMM)FROM EMP;

will only produce results for employees who have a non-null commission (or salary)

AGGREGATE OR GROUPING FUNCTIONS

AVG **AVG([DISTINCT]Column)**

This function returns the average of the values in the argument.

The data type of the argument must be numeric,date/time or character.The data type of the result is the same as the input argument.

DISTINCT eliminates duplicates.

e.g.Find the total salary budget for each department,the average salary,the number of people in each department.

```
SELECT emp.deptno,dname,SUM(sal),
AVG(sal),COUNT(empno)
    FROM emp,dept
    WHERE emp.deptno =dept.deptno
    GROUP BY emp.deptno,dname;

SELECT AVG(sal)“average”FROM emp;
```

COUNT **COUNT(*)**
(DISTINCT expression)

This function returns a count of items.

COUNT(*)always returns the number of rows in the table,rows that contain null values are included.

COUNT(column-name) returns the number of column values.

COUNT(DISTINCT column-name)filters out duplicate column values.

e.g.How many employees are in each department of the EMP table?

```
SELECT COUNT(*)FROM emp
    GROUPBY deptno;

SELECT COUNT(DISTINCTjob)"Jobs"FROM emp;
```

NOTE -the“”round Jobs is superfluous here,but must be used if the column heading is more than one word.