Table of Contents

In this discussion we will examine a few features of the Select statement. These are:

- selecting individual columns
- selecting all columns
- using column aliases
- selecting distinct rows
- sorting the rows displayed

1. Nulls

We can have an animal that does not have a name; the table was created with the z_name column being nullable. The following query inserts a row with an animal with no name. Note that the null value is not quoted in the values list.

Demo 01:

```
Insert Into zoo (z_id, z_name, z_type, z_cost, z_dob, z_acquired)
values  (99, null,'Horse', 490.00, '2010-05-15 08:30:00 ','2010-04-15');
```

2. Selecting columns

The first few queries use only two clauses: the FROM clause to identify the table that supplies the data and the SELECT clause to identify the columns to be returned. For these queries, all rows from the table are returned. This set of demos uses the zoo table. Your data set might be different depending on the rows you inserted.

You indicate which columns you want displayed and the order of the columns by listing the column names in the Select clause.

Demo 02: You can display the columns in any order. Note that row for the animal with no name displays the word NULL with this client.

```
Select
  z_type
, z_name
From zoo;
+-----------+----------------+
| z_type    | z_name         |
+-----------+----------------+
| Giraffe   | Sam            |
| Armadillo | Abigail        |
| Lion      | Leon           |
| Lion      | Lenora         |
| Giraffe   | Sally Robinson |
| Zebra     | Huey           |
| Zebra     | Dewey          |
| Zebra     | Louie          |
| Horse     | NULL           |
| giraffe   | Floyd          |
| giraffe   | Bri            |
| anteater  | NULL           |
| anteater  | NULL           |
| penguin   | Squeaky        |
| penguin   | Squack         |
| penguin   | Squeal         |
| penguin   | Squall         |
| penguin   | Squeaky        |
| anteater  | NULL           |
| anteater  | Baxter         |
+-----------+----------------+
20 rows in set (0.00 sec)
```

Demo 03: Display dates and numeric values.

```
Select
  z_dob
, z_cost
, z_name
From zoo;
+---------------------+---------+----------------+
| z_dob               | z_cost  | z_name         |
+---------------------+---------+----------------+
| 2002-05-15 10:45:00 | 5000.00 | Sam            |
| 2010-05-15 08:30:00 |  490.00 | Abigail        |
| 2009-02-25 15:00:00 | 5000.00 | Leon           |
| 2009-02-25 15:30:00 | 5000.00 | Lenora         |
| 2009-05-15 02:02:00 | 5000.25 | Sally Robinson |
| 2012-06-02 02:02:00 | 2500.25 | Huey           |
Partial output
```

3. Selecting all columns

The symbol * is used to indicate that all columns should be returned. This is inefficient if you do not need to see all of the columns but is helpful for a quick look at a small table.

Using Select * can be a bad idea with embedded SQL if the table design is changed. Embedded SQL refers to SQL statement that might be included inside other units of code. You also have to consider that someone might reorder the column positions in the table and then your query produces a different result.

Demo 04: Display all columns, all rows.

```
Select  *
From zoo;
+------+----------------+-----------+---------+---------------------+------------+
| z_id | z_name         | z_type    | z_cost  | z_dob               | z_acquired |
+------+----------------+-----------+---------+---------------------+------------+
|   23 | Sam            | Giraffe   | 5000.00 | 2002-05-15 10:45:00 | 2002-05-15 |
|   25 | Abigail        | Armadillo |  490.00 | 2010-05-15 08:30:00 | 2010-04-15 |
|   56 | Leon           | Lion      | 5000.00 | 2009-02-25 15:00:00 | 2011-01-15 |
|   57 | Lenora         | Lion      | 5000.00 | 2009-02-25 15:30:00 | 2011-01-15 |
|   85 | Sally Robinson | Giraffe   | 5000.25 | 2009-05-15 02:02:00 | 2012-03-15 |
|   43 | Huey           | Zebra     | 2500.25 | 2012-06-02 02:02:00 | 2012-06-02 |
Partial output
```

4. Column aliases

By default, the column headers are the attribute names. Column aliases can be used to supply different headers for the output display. You can display multiple columns using the same alias but it is not a good idea to do this.

Notice in the demos below how case issues are handled in the various ways of creating column aliases.

Demo 05: Display column headers other than the attribute names. The word AS is optional and may be omitted.

```
Select
  z_id
, z_dob   AS BirthDate
, z_cost  AS Price
, z_name  AS NAME
From zoo;
+------+---------------------+---------+----------------+
| z_id | BirthDate           | Price   | NAME           |
+------+---------------------+---------+----------------+
|   23 | 2002-05-15 10:45:00 | 5000.00 | Sam            |
|   25 | 2010-05-15 08:30:00 |  490.00 | Abigail        |
|   56 | 2009-02-25 15:00:00 | 5000.00 | Leon           |
|   57 | 2009-02-25 15:30:00 | 5000.00 | Lenora         |
|   85 | 2009-05-15 02:02:00 | 5000.25 | Sally Robinson |
Partial output
```

Demo 06: The use of double quotes for your aliases allows you to use spaces or special characters in the header.

```
Select
  z_id
, z_dob   AS "Date of Birth"
, z_cost  AS "Price $"
, z_name  As "Name"
From zoo;
+------+---------------------+---------+----------------+
| z_id | Date of Birth       | Price $ | Name           |
+------+---------------------+---------+----------------+
|   23 | 2002-05-15 10:45:00 | 5000.00 | Sam            |
|   25 | 2010-05-15 08:30:00 |  490.00 | Abigail        |
|   56 | 2009-02-25 15:00:00 | 5000.00 | Leon           |
|   57 | 2009-02-25 15:30:00 | 5000.00 | Lenora         |
|   85 | 2009-05-15 02:02:00 | 5000.25 | Sally Robinson |
|   43 | 2012-06-02 02:02:00 | 2500.25 | Huey           |
Partial output
```

5. Sorting the output display

If you want to control the order in which the rows are displayed, you use an ORDER BY clause.

You can order by

- a column
- a column alias
- the numeric position of the column in the Select (not always a good idea)
- a calculated column expression (we will discuss this in the next unit)

If you have two columns with the same alias and try to sort by the alias, you will get an error message.

Demo 07: Controlling the order in which the rows are displayed. This is sorted by price with the lower values first; this is an ascending sort which is the default sort order.

```
Select
  z_id
, z_dob   AS "BirthDate"
, z_cost  AS "Price"
, z_name  As "Name"
From zoo
ORDER BY z_cost;
+------+---------------------+---------+----------------+
| z_id | BirthDate           | Price   | Name           |
+------+---------------------+---------+----------------+
|   78 | 2003-12-12 08:45:00 |  400.00 | Squall         |
|   25 | 2010-05-15 08:30:00 |  490.00 | Abigail        |
|   99 | 2010-05-15 08:30:00 |  490.00 | NULL           |
|   78 | 2008-06-04 08:30:00 |  500.00 | Baxter         |
|   44 | 2006-02-22 02:02:00 |  500.00 | NULL           |
|   12 | 2006-02-22 00:00:00 |  500.00 | NULL           |
|   75 | 2003-12-12 00:00:00 |  500.00 | Squeaky        |
|   79 | 2008-10-08 08:45:00 |  500.00 | Squeaky        |
|   77 | 2008-04-01 05:00:00 |  500.00 | NULL           |
|   90 | 2009-01-26 02:02:00 |  500.00 | Bri            |
|   77 | 2005-12-12 08:45:00 |  750.00 | Squeal         |
|   76 | 2005-12-10 00:00:00 |  750.00 | Squack         |
|   45 | 2013-01-02 02:25:00 | 2500.25 | Louie          |
|   44 | 2012-06-02 02:10:00 | 2500.25 | Dewey          |
|   43 | 2012-06-02 02:02:00 | 2500.25 | Huey           |
|   52 | 2003-05-15 15:00:00 | 3500.00 | Floyd          |
|   57 | 2009-02-25 15:30:00 | 5000.00 | Lenora         |
|   56 | 2009-02-25 15:00:00 | 5000.00 | Leon           |
|   23 | 2002-05-15 10:45:00 | 5000.00 | Sam            |
|   85 | 2009-05-15 02:02:00 | 5000.25 | Sally Robinson |
+------+---------------------+---------+----------------+
20 rows in set (0.00 sec)
```

Demo 08: Using DESC to specify a descending sort.

```
Select
  z_id
, z_dob   AS "BirthDate"
, z_cost  AS "Price"
, z_name  As "Name"
From zoo
ORDER BY z_cost DESC;
+------+---------------------+---------+----------------+
| z_id | BirthDate           | Price   | Name           |
+------+---------------------+---------+----------------+
|   85 | 2009-05-15 02:02:00 | 5000.25 | Sally Robinson |
|   23 | 2002-05-15 10:45:00 | 5000.00 | Sam            |
|   56 | 2009-02-25 15:00:00 | 5000.00 | Leon           |
|   57 | 2009-02-25 15:30:00 | 5000.00 | Lenora         |
|   52 | 2003-05-15 15:00:00 | 3500.00 | Floyd          |
|   45 | 2013-01-02 02:25:00 | 2500.25 | Louie          |
. . .  rows ommitted
```

```
|   99 | 2010-05-15 08:30:00 |  490.00 | NULL           |
|   25 | 2010-05-15 08:30:00 |  490.00 | Abigail        |
|   78 | 2003-12-12 08:45:00 |  400.00 | Squall         |
+------+---------------------+---------+----------------+
20 rows in set (0.00 sec)
```

Demo 09: This is a two level sort. The first sort key is the z_type. If the z_type values of two rows match, then the z_cost value is used for the second sort level.

```
Select
  z_type  As "Type"
, z_cost  AS "Price"
, z_name  As "Name"
From zoo
ORDER BY z_type, z_cost;
+-----------+---------+----------------+
| Type      | Price   | Name           |
+-----------+---------+----------------+
| anteater  |  500.00 | Baxter         |
| anteater  |  500.00 | NULL           |
| anteater  |  500.00 | NULL           |
| anteater  |  500.00 | NULL           |
| Armadillo |  490.00 | Abigail        |
| giraffe   |  500.00 | Bri            |
| giraffe   | 3500.00 | Floyd          |
. . .  rows ommitted
| penguin   |  400.00 | Squall         |
| penguin   |  500.00 | Squeaky        |
| penguin   |  500.00 | Squeaky        |
| penguin   |  750.00 | Squack         |
| penguin   |  750.00 | Squeal         |
. . .  rows ommitted
+-----------+---------+----------------+
20 rows in set (0.00 sec)
```

Demo 10: This is a two level sort. The first sort key is the z_type and it is ascending. The second sort key z_cost uses a descending sort.

```
Select
  z_type  As "Type"
, z_cost  AS "Price"
, z_name  As "Name"
From zoo
ORDER BY z_type, z_cost desc;
+-----------+---------+----------------+
| Type      | Price   | Name           |
+-----------+---------+----------------+
| anteater  |  500.00 | Baxter         |
| anteater  |  500.00 | NULL           |
| anteater  |  500.00 | NULL           |
| anteater  |  500.00 | NULL           |
| Armadillo |  490.00 | Abigail        |
| Giraffe   | 5000.25 | Sally Robinson |
| Giraffe   | 5000.00 | Sam            |
| giraffe   | 3500.00 | Floyd          |
| giraffe   |  500.00 | Bri            |
Partial output
```

Demo 11: The default is that nulls sort as a low-valued data item. We have animals with no name value. They are sorting at the top of this display.

```
Select
  z_type  As "Type"
, z_name  As "Name"
From zoo
ORDER BY z_name;
+-----------+----------------+
| Type      | Name           |
+-----------+----------------+
| anteater  | NULL           |
| anteater  | NULL           |
| anteater  | NULL           |
| Horse     | NULL           |
| Armadillo | Abigail        |
| anteater  | Baxter         |
| giraffe   | Bri            |
| Zebra     | Dewey          |
| giraffe   | Floyd          |
Partial output
```

Demo 12: With a Desc z_name sort the nulls are at the end of the result set.

```
Select
  z_type  As "Type"
, z_name  As "Name"
From zoo
ORDER BY z_name DESC;
+-----------+----------------+
| Type      | Name           |
+-----------+----------------+
| penguin   | Squeal         |
| penguin   | Squeaky        |
| penguin   | Squeaky        |
| penguin   | Squall         |
| penguin   | Squack         |
. . .  rows ommitted
| anteater  | NULL           |
| anteater  | NULL           |
| Horse     | NULL           |
| anteater  | NULL           |
+-----------+----------------+
20 rows in set (0.00 sec)
```

Demo 13: You can sort on a date value.

```
Select
  z_id
, z_dob   as "BirthDate"
, z_name  as "Name"
From zoo
ORDER BY z_dob DESC;
+------+---------------------+----------------+
| z_id | BirthDate           | Name           |
+------+---------------------+----------------+
|   45 | 2013-01-02 02:25:00 | Louie          |
|   44 | 2012-06-02 02:10:00 | Dewey          |
|   43 | 2012-06-02 02:02:00 | Huey           |
|   25 | 2010-05-15 08:30:00 | Abigail        |
|   99 | 2010-05-15 08:30:00 | NULL           |
|   85 | 2009-05-15 02:02:00 | Sally Robinson |
|   57 | 2009-02-25 15:30:00 | Lenora         |
|   56 | 2009-02-25 15:00:00 | Leon           |
Partial output
```

Demo 14: You can sort by a column alias. Since this alias includes spaces, it needs to be quoted and you need to use the back tick.

```
Select
  z_id
, z_dob   as "Date of Birth"
, z_name  as "Name"
From zoo
ORDER BY `Date of Birth`;
+------+---------------------+----------------+
| z_id | Date of Birth       | Name           |
+------+---------------------+----------------+
|   23 | 2002-05-15 10:45:00 | Sam            |
|   52 | 2003-05-15 15:00:00 | Floyd          |
|   75 | 2003-12-12 00:00:00 | Squeaky        |
|   78 | 2003-12-12 08:45:00 | Squall         |
|   76 | 2005-12-10 00:00:00 | Squack         |
|   77 | 2005-12-12 08:45:00 | Squeal         |
|   12 | 2006-02-22 00:00:00 | NULL           |
|   44 | 2006-02-22 02:02:00 | NULL           |
|   77 | 2008-04-01 05:00:00 | NULL           |
Partial output
```

Demo 15: What happens if you use double quotes on the sort key identifier? Are these rows sorted in date order?

```
Select
  z_id
, z_dob   as "Date of Birth"
, z_name  as "Name"
From zoo
ORDER BY "Date of Birth";
+------+---------------------+----------------+
| z_id | Date of Birth       | Name           |
+------+---------------------+----------------+
|   23 | 2002-05-15 10:45:00 | Sam            |
|   25 | 2010-05-15 08:30:00 | Abigail        |
|   56 | 2009-02-25 15:00:00 | Leon           |
|   57 | 2009-02-25 15:30:00 | Lenora         |
|   85 | 2009-05-15 02:02:00 | Sally Robinson |
|   43 | 2012-06-02 02:02:00 | Huey           |
|   44 | 2012-06-02 02:10:00 | Dewey          |
|   45 | 2013-01-02 02:25:00 | Louie          |
|   99 | 2010-05-15 08:30:00 | NULL           |
|   52 | 2003-05-15 15:00:00 | Floyd          |
|   90 | 2009-01-26 02:02:00 | Bri            |
|   44 | 2006-02-22 02:02:00 | NULL           |
|   12 | 2006-02-22 00:00:00 | NULL           |
|   75 | 2003-12-12 00:00:00 | Squeaky        |
|   76 | 2005-12-10 00:00:00 | Squack         |
|   77 | 2005-12-12 08:45:00 | Squeal         |
|   78 | 2003-12-12 08:45:00 | Squall         |
|   79 | 2008-10-08 08:45:00 | Squeaky        |
|   77 | 2008-04-01 05:00:00 | NULL           |
|   78 | 2008-06-04 08:30:00 | Baxter         |
+------+---------------------+----------------+
20 rows in set (0.00 sec)
```

Demo 16: MySQL allows you to sort by the column number. This is not generally considered good style since it is easy to rearrange the column in the select and forget to adjust the Order By clause. You want to write SQL that is easier to write correctly and harder to write incorrectly.
This will sort by the z_type values then by the z_name values.

```
Select
  z_id
, z_type
, z_name
From zoo
ORDER BY 2,3;
+------+-----------+----------------+
| z_id | z_type    | z_name         |
+------+-----------+----------------+
|   77 | anteater  | NULL           |
|   12 | anteater  | NULL           |
|   44 | anteater  | NULL           |
|   78 | anteater  | Baxter         |
|   25 | Armadillo | Abigail        |
|   90 | giraffe   | Bri            |
|   52 | giraffe   | Floyd          |
Partial output
```

You can sort on calculated columns, either by using the alias or repeating the calculation as the sort key. We discuss calculation later; this is included here for completeness. Extract (month..) gives us the numerical value of the month.

Demo 17:

```
Select z_id
, extract( Month from z_dob)   AS "Birth Month"
, z_name  As "Name"
From zoo
ORDER BY extract( Month from z_dob);
+------+-------------+----------------+
| z_id | Birth Month | Name           |
+------+-------------+----------------+
|   90 |           1 | Bri            |
|   45 |           1 | Louie          |
|   56 |           2 | Leon           |
|   57 |           2 | Lenora         |
|   12 |           2 | NULL           |
|   44 |           2 | NULL           |
|   77 |           4 | NULL           |
|   23 |           5 | Sam            |
|   52 |           5 | Floyd          |
|   99 |           5 | NULL           |
|   25 |           5 | Abigail        |
|   85 |           5 | Sally Robinson |
|   78 |           6 | Baxter         |
|   43 |           6 | Huey           |
|   44 |           6 | Dewey          |
|   79 |          10 | Squeaky        |
|   75 |          12 | Squeaky        |
|   77 |          12 | Squeal         |
|   78 |          12 | Squall         |
|   76 |          12 | Squack         |
+------+------------+----------------+
20 rows in set (0.00 sec)
```

6. Selecting distinct output rows

The keyword DISTINCT can be placed after SELECT to specify that any duplicate copies of an output row will not be displayed. The decision is based on the uniqueness of the rows to be displayed, not the uniqueness of rows in the table.

Using DISTINCT takes extra processing time and should be avoided unless it is necessary to avoid duplicate output lines. If you are writing a query that uses a single table and displays the primary key, do not include DISTINCT.

Demo 18: Display one output row per row in the table

```
Select z_type
From zoo;
+-----------+
| z_type    |
+-----------+
| Giraffe   |
| Armadillo |
| Lion      |
| Lion      |
| Giraffe   |
| Zebra     |
| Zebra     |
| Zebra     |
| Horse     |
| giraffe   |
| giraffe   |
| anteater  |
| anteater  |
| penguin   |
| penguin   |
| penguin   |
| penguin   |
| penguin   |
| anteater  |
| anteater  |
+-----------+
20 rows in set (0.00 sec)
```

Demo 19: Display one output row for each different value of z_type. We have three zebras in the table but with Distinct we get only one row for zebra.

```
Select DISTINCT z_type
From zoo;
+-----------+
| z_type    |
+-----------+
| Giraffe   |
| Armadillo |
| Lion      |
| Zebra     |
| Horse     |
| anteater  |
| penguin   |
+-----------+
7 rows in set (0.00 sec)
```

Demo 20: Display one output row for each different combination of values for z_type and z_cost. We have two rows with Giraffe because there are two different price values for giraffes.

```
Select DISTINCT z_type, z_cost
From zoo
ORDER BY z_type, z_cost;
+-----------+---------+
| z_type    | z_cost  |
+-----------+---------+
| anteater  |  500.00 |
| Armadillo |  490.00 |
| giraffe   |  500.00 |
| giraffe   | 3500.00 |
| Giraffe   | 5000.00 |
| Giraffe   | 5000.25 |
| Horse     |  490.00 |
| Lion      | 5000.00 |
| penguin   |  400.00 |
| penguin   |  500.00 |
| penguin   |  750.00 |
| Zebra     | 2500.25 |
+-----------+---------+
12 rows in set (0.00 sec)
```

Please note that Distinct is not a function and it is inappropriate to use parentheses with Distinct. You will often see queries that use the syntax Select Distinct (z_type) from zoo, but those parentheses are simply parentheses you could use around any expression. You can write a query such as this where the parentheses are also legal but meaningless. Select (z_type) from zoo;

6.1. Sorting and distinct

Return to the query

```
Select DISTINCT z_type
from zoo;
```

Can we sort the output? With some dbms, the way a Distinct operation is implemented, the output is commonly sorted.

Demo 21: Distinct Z-Type, order by z_type

```
Select DISTINCT z_type
From zoo
order by z_type;
+-----------+
| z_type    |
+-----------+
| anteater  |
| Armadillo |
| Giraffe   |
| Horse     |
| Lion      |
| penguin   |
| Zebra     |
+-----------+
7 rows in set (0.00 sec)
```

What if we want to sort the output by the animal name? Before we try this, think about what this means. We are displaying one row that represents all of the zebras, one row that represents all of the giraffes. If we sort by z_name, how should the rows be returned? MySQL allows an order by clause to sort by the name even if it is not in the Select**. This is a MySQL extension and is not allowed in all dbms**.

Demo 22:

```
Select DISTINCT z_type
From zoo
order by z_name;
+-----------+
| z_type    |
+-----------+
| anteater  |
| Horse     |
| Armadillo |
| Zebra     |
| Lion      |
| Giraffe   |
| penguin   |
+-----------+
```

MySQL allows some extensions for the purpose of improving efficiency of retrieval. (Every dbms does this.) You need to decide (1) if this sort makes sense and is useful, and (2) if you want to keep your SQL closer to standard SQL. There is nothing wrong with using the MySQL extensions but it helps to be aware of them.