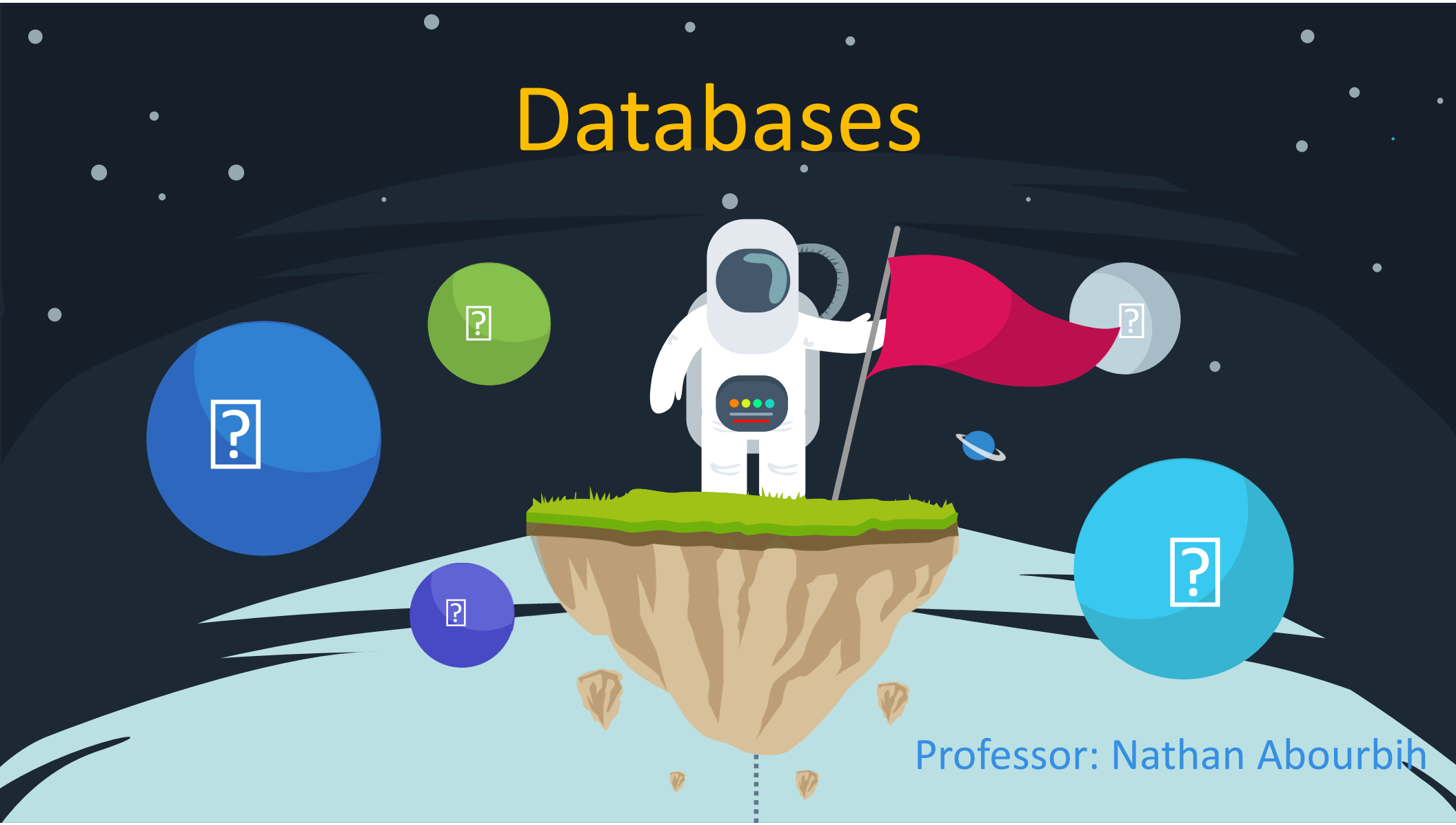
Databases



Professor: Nathan Abourbih

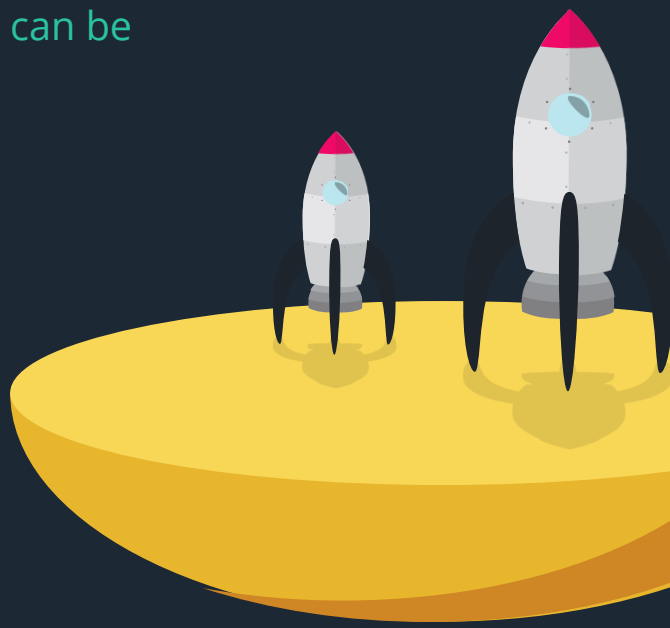
Data Types

A column's data type specifies the kind of information the column is intended to store.

In addition, a column's data type determines the operations that can be performed on the column.

Data types can be divided into the five categories shown below:

Category	Description
Character	Strings of character data
Numeric	Numbers that don't include a decimal point (integers) and numbers that include a decimal point (real numbers)
Date and time	Dates, times, or both
Large Object (LOB)	Large strings of character or binary data
Spatial	Geographical values
JSON	JSON documents



Introduction to Data Types

Character data types are intended for storing a string of one or more characters, which can include letters, numbers, symbols, or special characters.

The terms character, string, and text are used interchangeably to describe this type of data.

Date and time data types are intended for storing dates, times, or both dates and times.

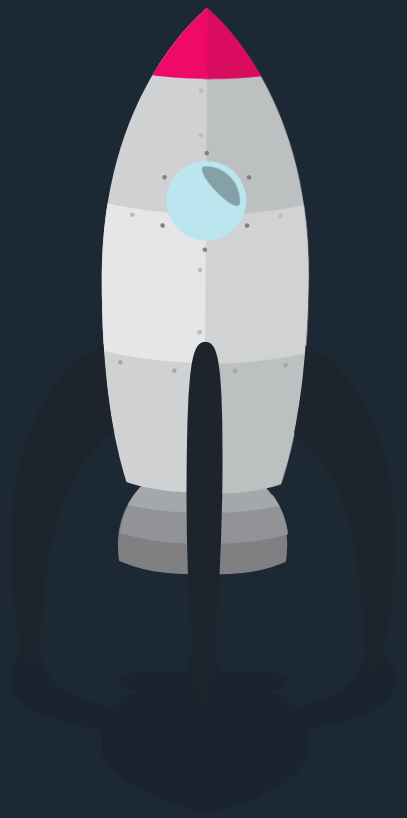
Introduction to Data Types

Numeric data types are intended for storing numbers that can be used for mathematical calculations.

At a basic level, you can divide numbers into two categories:

Integers are numbers that don't have a decimal point

Real Numbers are numbers that have a decimal point.



Introduction to Data Types

- Numbers that **don't include a decimal** point are known as integers.
- Numbers that **include a decimal** point are known as real numbers.
- The **date and time** data types are often referred to as the date/time or temporal data types.
- The **large object** (LOB) data types are useful for storing images, sound, video, and large amounts of text.
- The **spatial** data types are useful for storing geometric or geographical values such as global positioning system (GPS) data. These data types are referred to as geometry types.
- The **JSON** data type is used for storing JavaScript Object Notation (JSON) documents.

Category	Description
Character	Strings of character data
Numeric	Numbers that don't include a decimal point (integers) and numbers that include a decimal point (real numbers)
Date and time	Dates, times, or both
Large Object (LOB)	Large strings of character or binary data
Spatial	Geographical values
JSON	JSON documents

Character Types

The CHAR type is used for fixed-length strings.

A column with this type uses the same amount of storage for each value regardless of the actual length of the string.

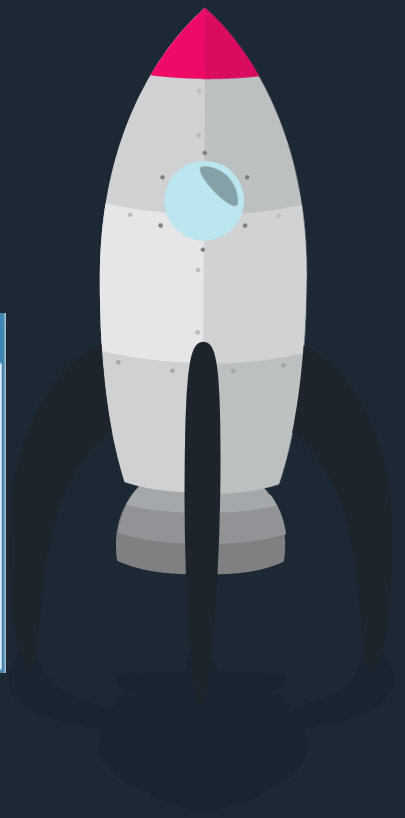
The VARCHAR type is used for variable-length strings.

A column with this type uses a varying amount of storage for each value depending on the length of the string.

Character Types

Type	Bytes	Description
`CHAR(M)`	Mx4	Fixed-length strings of character data where M is the number of characters, between 0 and 255. With the utf8mb4 character set, MySQL must reserve four bytes for each character in a CHAR column because that's the maximum possible length.
`VARCHAR(M)`	L+1	Variable-length strings of character data where M is the maximum number of characters, between 0 and 255. For English and Latin characters, the number of bytes used to store the string is equal to length of the string (L) plus 1 byte to record its length.

Data type	Original value	Value stored	Bytes used
`CHAR(2)`	`'CA'`	`'CA'`	8
`CHAR(10)`	`'CA'`	`'CA        '`	40
`VARCHAR(10)`	`'CA'`	`'CA'`	3
`VARCHAR(20)`	`'California'`	`'California'`	11
`VARCHAR(20)`	`'New York'`	`'New York'`	9
`VARCHAR(20)`	`"Murach's MySQL"`	`"Murach's MySQL"`	15



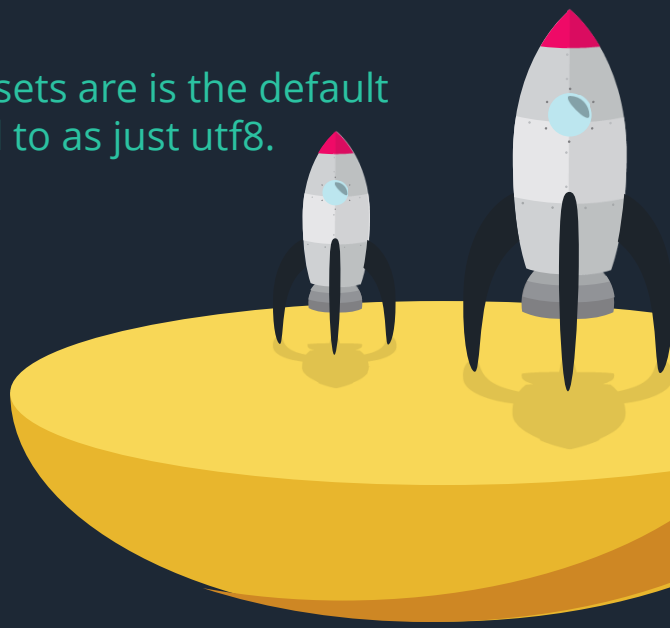
Character Sets

When a column is defined with a string type such as CHAR or VARCHAR, MySQL stores a numeric value for each character.

Then, it uses a character set to map the numeric values to the characters of the string.

The utf8mb3/utf8mb4 character sets are is the default for MySQL and are often referred to as just utf8.

Name	Description
`latin1`	The latin1 character set uses one byte per character to provide for most characters in Western European languages.
`utf8mb3`	The utf8mb3 character set uses one to three bytes per character to provide for all characters specified by the Unicode character set. This character set provides for most characters in most of the world's languages.
`utf8mb4`	The utf8mb4 character set uses one to four bytes per character to provide for all characters specified by the Unicode character set, plus additional characters like emojis.



Integer Types

The integer types store numbers without any digits to the right of the decimal point.

Type	Bytes	Value ranges
`BIGINT`	8	Signed: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Unsigned: 0 to 18,446,744,073,709,551,615
`INT`	4	Signed: -2,147,483,648 to 2,147,483,647 Unsigned: 0 to 4,294,967,295
`MEDIUMINT`	3	Signed: -8,388,608 5to 8,388,607 Unsigned: 0 and 16,777,215
`SMALLINT`	2	Signed: -32,768 and 32,767 Unsigned: 0 and 65,535
`TINYINT`	1	Signed: -128 and 127 Unsigned: 0 and 255

Integer Types – Signed VS Unsigned

If the UNSIGNED attribute for the integer is set, it changes the range of acceptable values. If you try to store a negative integer in a column with the UNSIGNED attribute, an error occurs.

Data type	Original value	Value stored	Value displayed
INT	99	99	99
INT	-99	-99	-99
INT UNSIGNED	99	99	99
INT UNSIGNED	-99	None	None

Integer Types – Zero Fill

If the ZEROFILL attribute for the integer is set, MySQL displays the integer with zeros padded from the left, up to the maximum display size.

If the ZEROFILL attribute is set, MySQL automatically sets the UNSIGNED attribute.

Data type	Original value	Value stored	Value displayed
INT ZEROFILL	99	99	0000000099
INT(4) ZEROFILL	99	99	0099

Real Numbers - Exact

Real numbers can include digits to the right of the decimal point.

The precision of a real number indicates the total number of digits that can be stored, and the scale indicates the number of digits that can be stored to the right of the decimal point.

DECIMAL(max_digits, decimals)

Type	Bytes	Description
`DECIMAL(M, D)`	Vary	Fixed-precision numbers where M specifies the maximum number of total digits (the precision) and D specifies the number of digits to the right of the decimal (the scale). M can range from 1 to 65. D can range from 0 to 30 but can't be larger than M. The default is 0.

The DECIMAL type is considered an exact numeric type because its precision is exact.

If the UNSIGNED attribute for a real number is set, it prevents storing negative values in the column but does not affect the range of acceptable values

Real Numbers – Floating-Point

Real numbers can include digits to the right of the decimal point.

The precision of a real number indicates the total number of digits that can be stored, and the scale indicates the number of digits that can be stored to the right of the decimal point.

Type	Bytes	Description
DOUBLE	8	Double-precision floating-point numbers from -1.7976×10^{308} to 1.7976×10^{308}.
FLOAT	4	Single-precision floating-point numbers from -3.4028×10^{38} to 3.4028×10^{38}.

The DOUBLE and FLOAT types store floating-point numbers, which have a limited number of significant digits. These data types are considered approximate numeric data types because they may not represent a value exactly.

For business applications, you typically use the exact numeric types, as there's seldom the need to work with the very large and very small numbers that the floating-point data types are designed for. However, for scientific applications, you may sometimes need to use the DOUBLE and FLOAT types.

Date/Time Types

Type	Bytes	Description
DATE	3	Dates from January 1, 1000 through December 31, 9999. The default format for display and entry is “yyyy-mm-dd”.
TIME	3	Times in the range -838:59:59 through 838:59:59. The default format for display and entry is “hh:mm:ss”.
DATETIME	8	Combination date and time from midnight January 1, 1970 to December 31, 9999. The default format for display and entry is “yyyy-mm-dd hh:mm:ss”.
TIMESTAMP	4	Combination date and time from midnight January 1, 1970 to the year 2037. The default format is “yyyy-mm-dd hh:mm:ss”.
YEAR[(4)]	1	Years in 4-digit format. Allowable values are from 1901 to 2155.

Date/Time Interpretation

- You can specify date and time values by coding a literal value. In most cases, you enclose the literal value in single quotes.
- For dates, MySQL uses the “yyyy-mm-dd” format. For times, MySQL uses the “hh:mm:ss” format, using a 24-hour clock.
- If you don’t specify a time when storing a DATETIME or TIMESTAMP value, MySQL stores a time value of 00:00:00 (12:00 midnight).

Literal value	Value stored in DATE column
'2018-08-15'	2018-08-15
'2018-8-15'	2018-08-15
'18-8-15'	2018-08-15
'20180815'	2018-08-15
20180815	2018-08-15
'2018.08.15'	2018-08-15
'18/8/15'	2018-08-15
'8/15/18'	None
'2018-02-31'	None
Literal value	**Value stored in TIME column**
'7:32'	07:32:00
'19:32:11'	19:32:11
'193211'	19:32:11
193211	19:32:11
'19:61:11'	None
Literal value	**Value stored in DATETIME or TIMESTAMP column**
'2018-08-15 19:32:11'	2018-08-15 19:32:11
'2018-08-15'	2018-08-15 00:00:00

ENUM and SET

Type	Bytes	Description
`ENUM`	1-2	Stores one value selected from a list of acceptable values.
`SET`	1-8	Stores zero or more values selected from a list of acceptable values.

Value	Stored in column ENUM ('Yes', 'No', 'Maybe')
`'Yes'`	`'Yes'`
`'No'`	`'No'`
`'Maybe'`	`'Maybe'`
`'Possibly'`	`''`

Value	Stored in column SET ('Pepperoni', 'Mushrooms', 'Olives')
`'Pepperoni'`	`'Pepperoni'`
`'Mushrooms'`	`'Mushrooms'`
`'Pepperoni, Bacon'`	`'Pepperoni'`
`'Olives, Pepperoni'`	`'Pepperoni, Olives'`

ENUM and SET Details

- The **ENUM** and **SET** types can be used to restrict the values that you store to a limited set of values. The ENUM column can take on exactly one value, but a SET column can take on zero, one, or up to 64 different values.
- You can define the set of acceptable values for an **ENUM** or **SET** column when you create a table. An **ENUM** column can have up to 65,535 acceptable values, but a **SET** column is limited to 64 acceptable values.
- To specify a value for an **ENUM** column, you code a single text string. If the string contains an acceptable value, that value is stored in the column. Otherwise, the column is assigned an empty string.
- If you don't specify a value for an **ENUM** column when you insert a row, MySQL assigns a default value that depends on whether the column allows null values. If the column allows null values, MySQL assigns a null value to the column. If it doesn't allow null values, MySQL assigns the first value in the set of acceptable values to the column.
- To specify values for a **SET** column, you code a single string with the values separated by commas. Each acceptable value is stored in the column, and any other values are ignored.
- When you store values in a **SET** column, MySQL stores the values using the order specified in the column definition, and it does not store duplicate values.

Large Object Types

Type	Bytes	Description
LONGBLOB	L+4	Variable-length strings of binary data up to 4GB in length (L).
MEDIUMBLOB	L+3	Variable-length strings of binary data up to 16MB in length (L).
BLOB	L+2	Variable-length strings of binary data up to 65KB in length (L).
TINYBLOB	L+1	Variable-length strings of binary data up to 255 bytes in length (L).
LONGTEXT	L+4	Variable-length strings of characters up to 4GB in length (L).
MEDIUMTEXT	L+3	Variable-length strings of characters up to 16MB in length (L).
TEXT	L+2	Variable-length strings of characters up to 65KB in length (L).
TINYTEXT	L+1	Variable-length strings of characters up to 255 bytes in length (L).

The BLOB types store strings of binary data and are referred to as binary large object (BLOB) types.

The TEXT types store strings of character data and are sometimes referred to as character large object (CLOB) types.

Implicitly convert a number to a string

```
SELECT
    invoice_total
    , CONCAT('$', invoice_total)
FROM
    invoices
```

invoice_total	CONCAT('$', invoice_total)
3813.33	$3813.33
40.20	$40.20

Implicitly convert a string to a number

```
SELECT
    invoice_number
    , 989319/invoice_number
FROM
    invoices
```

invoice_number	989319/invoice_number
989319-457	1
263253241	0.0037580505988908225
963253234	0.0010270601385803393

Implicitly convert a date to a number

```
SELECT
    invoice_date,
    invoice_date + 1
FROM
    invoices
```

invoice_date	invoice_date + 1
2014-04-08	20140409
2014-04-10	20140411

The syntax of the CAST function

```
CAST(expression AS cast_type)
```

The syntax of the CONVERT function

```
CONVERT(expression, cast_type)
```

Cast types you can use in these functions

```
CHAR[(N)]
DATE
DATETIME
TIME
SIGNED [INTEGER]
UNSIGNED [INTEGER]
DECIMAL[(M[,D])]
```

A statement that uses the CAST function

```
SELECT
    invoice_id
    , invoice_date
    , invoice_total
    , CAST(invoice_date AS CHAR(10)) AS char_date
    , CAST(invoice_total AS SIGNED) AS integer_total
FROM
    invoices
```

invoice_id	invoice_date	invoice_total	char_date	integer_total
1	2014-04-08	3813.33	2014-04-08	3813
2	2014-04-10	40.20	2014-04-10	40
3	2014-04-13	138.75	2014-04-13	139

A statement that uses the CONVERT function

```
SELECT
    invoice_id
    , invoice_date
    , invoice_total
    , CONVERT(invoice_date, CHAR(10)) AS char_date
    , CONVERT(invoice_total, SIGNED) AS integer_total
FROM
    invoices
```

invoice_id	invoice_date	invoice_total	char_date	integer_total
1	2014-04-08	3813.33	2014-04-08	3813
2	2014-04-10	40.20	2014-04-10	40
3	2014-04-13	138.75	2014-04-13	139

The FORMAT and CHAR functions

```
FORMAT(number,decimal)
CHAR(value1[,value2]...)
```

FORMAT function examples

Function	Result
`FORMAT(1234567.8901,2)`	`1,234,567.89`
`FORMAT(1234.56,4)`	`1,234.5600`
`FORMAT(1234.56,0)`	`1,235`

Function	Description
`FORMAT(number,decimal)`	Converts the specified number to a character string with grouped digits separated by commas, rounded to the specified number of decimal digits. If decimal is zero, then the decimal point is omitted.
`CHAR(value1[,value2]...)`	Converts one or more numbers to a binary string. Each number is interpreted as an integer between 0 and 255.

CHAR function examples for common control characters

Function	Control character
`CHAR(9)`	Tab
`CHAR(10)`	Line feed
`CHAR(13)`	Carriage return

```
SELECT
    CONCAT(
        vendor_name, CHAR(13,10)
        , vendor_address1, CHAR(13,10)
        , vendor_city, ', ', vendor_state, ' ', vendor_zip_code
    )
FROM
    vendors
WHERE
    vendor_id = 1;
```

```
US Postal Service
Attn:  Supt. Window Services
Madison, WI 53707
```

TRY IT YOURSELF

MYSQL – DATA TYPES

Using the ap database, write a query to satisfy the following requirements:

1. Write a SELECT statement that returns these columns from the Invoices table:

   The invoice_total column

   A column that uses the FORMAT function to return the invoice_total column with 1 digit to the right of the decimal point

   A column that uses the CONVERT function to return the invoice_total column as an integer

   A column that uses the CAST function to return the invoice_total column as an integer

	invoice_total	total_format	total_convert	total_cast
▶	3813.33	3,813.3	3813	3813
	40.20	40.2	40	40
	138.75	138.8	139	139
	144.70	144.7	145	145
	15.50	15.5	16	16
	42.75	42.8	43	43
	172.50	172.5	173	173

TRY IT YOURSELF

MYSQL – DATA TYPES

Using the ap database, write a query to satisfy the following requirements:

2. Write a SELECT statement that returns these columns from the Invoices table:

   The invoice_date column

   A column that uses the CAST function to return the invoice_date column with its full date and time

   A column that uses the CAST function to return the invoice_date column with just the year and the month

	invoice_date	invoice_datetime	invoice_char
▶	2018-08-02	2018-08-02 00:00:00	2018-08
	2018-08-01	2018-08-01 00:00:00	2018-08
	2018-07-31	2018-07-31 00:00:00	2018-07
	2018-07-30	2018-07-30 00:00:00	2018-07
	2018-07-28	2018-07-28 00:00:00	2018-07
	2018-07-25	2018-07-25 00:00:00	2018-07

REVIEW: Applied Activity 1

Write a SQL query that satisfies the requirements below.

Requirements

- Use the database my_guitar_shop.
- SELECT the rows from the products table.
- Return a resultset that matches the results shown below.

Results

	id	name	price	discount
▶	3	Gibson SG	2517.00	52.00
	2	Gibson Les Paul	1199.00	30.00
	7	Fender Precision	799.99	30.00
	10	Tama 5-Piece Drum Set with Cymbals	799.99	15.00
	9	Ludwig 5-piece Drum Set with Cymbals	699.99	30.00
	1	Fender Stratocaster	699.00	30.00
	8	Hofner Icon	499.99	25.00
	4	Yamaha FG700S	489.99	38.00
	6	Rodriguez Caballero 11	415.00	39.00
	5	Washburn D10S	299.00	0.00

```
SELECT
    product_id AS id
    , product_name AS name
    , list_price AS price
    , discount_percent as discount
FROM
    products
ORDER BY
    list_price DESC
```

REVIEW: Applied Activity 1

Write a SQL query that satisfies the requirements below.

Requirements

- Use the database my_guitar_shop.
- SELECT the rows from the customers table.
- Only include customers that have a last name that starts with a letter of the alphabet that comes after the letter N.
- Return a resultset that matches the results shown below.

Results

	customer_name
▶	Zimmer, Barry
	Wilson, Frank Lee
	Valentino, Erin
	Sherwood, Allan

```sql
SELECT
    CONCAT(last_name, ', ', first_name) AS customer_name
FROM
    customers
WHERE
    last_name > 'N'
ORDER BY
    last_name DESC
```

REVIEW: Applied Activity 1

Write a SQL query that satisfies the requirements below.

Requirements

- Use the database my_guitar_shop.
- SELECT the rows from the products table.
- Only include products that have a list price between seven hundred and fifty dollars and fifteen hundred dollars.
- Return a resultset that matches the results shown below.

Results

	name	added	price
▶	Gibson Les Paul	2011-12-05 16:33:13	1199.00
	Fender Precision	2012-06-01 11:29:35	799.99
	Tama 5-Piece Drum Set with Cymbals	2012-07-30 13:14:15	799.99

```sql
SELECT
    product_name AS name
    , date_added AS added
    , list_price AS price
FROM
    products
WHERE
    list_price BETWEEN 750 AND 1500
ORDER BY
    date_added ASC
```

REVIEW: Applied Activity 1

Write a SQL query that satisfies the requirements below.

Requirements

- Use the database my_guitar_shop.
- SELECT the rows from the products table.
- Return the following columns and data:
  - product name — The product_name column.
  - list price — The list_price column.
  - discount percent — The discount_percent column.
  - discount amount — A column that's calculated from the previous two columns.
  - discount price — A column that's calculated from the previous three columns.
- Round the discount amount and discount price columns to 2 decimal places.
- Sort the resultset by discount price in ascending sequence.
- Use the LIMIT clause so the result set contains only the first 6 rows.
- Return a resultset that matches the results shown below.

```sql
SELECT
    product_name AS 'Product Name'
    , list_price AS 'List Price'
    , discount_percent
        AS 'Discount Percent'
    , ROUND(list_price * discount_percent * .01, 2)
        AS 'Discount Amount'
    , list_price - ROUND(list_price * discount_percent * .01, 2)
        AS 'Discount Price'
FROM
    products
ORDER BY
    `Discount Price`  ASC
LIMIT 6
```

Results

Product Name	List Price	Discount Percent	Discount Amount	Discount Price
Rodriguez Caballero 11	415.00	39.00	161.85	253.15
Washburn D10S	299.00	0.00	0.00	299.00
Yamaha FG700S	489.99	38.00	186.20	303.79
Hofner Icon	499.99	25.00	125.00	374.99
Fender Stratocaster	699.00	30.00	209.70	489.30
Ludwig 5-piece Drum Set with Cymbals	699.99	30.00	210.00	489.99

REVIEW: Applied Activity 1

Write a SQL query that satisfies the requirements below.

Requirements

- Use the database my_guitar_shop.
- SELECT the rows from the orders table.
- Return a resultset that matches the results shown below.

Results

	Order	Customer	Amount	Ordered Date	Shipped Date
▶	1	1	5.00	2012-03-28 09:40:28	2012-03-30 15:32:51
	2	2	5.00	2012-03-28 11:23:20	2012-03-29 12:52:14
	3	1	10.00	2012-03-29 09:44:58	2012-03-31 09:11:41
	4	3	5.00	2012-03-30 15:22:31	2012-04-03 16:32:21
	5	4	5.00	2012-03-31 05:43:11	2012-04-02 14:21:12
	7	6	15.00	2012-04-01 23:11:12	2012-04-03 10:21:35

```
SELECT
    order_id AS 'Order'
    , customer_id AS 'Customer'
    , ship_amount AS 'Amount'
    , order_date AS 'Ordered Date'
    , ship_date AS 'Shipped Date'
FROM
    orders
WHERE
    ship_date IS NOT NULL
```

REVIEW: Applied Activity 1

Write a SQL query that satisfies the requirements below.

Requirements

- Use the database my_guitar_shop.
- SELECT the rows from the orders table.
- The second column should make use of the date format function.
- Return a resultset that matches the results shown below.

Results

	Order Date	Order Date Formatted
▶	2012-03-28 09:40:28	28-Mar-2012
	2012-03-28 11:23:20	28-Mar-2012
	2012-03-29 09:44:58	29-Mar-2012
	2012-03-30 15:22:31	30-Mar-2012
	2012-03-31 05:43:11	31-Mar-2012
	2012-03-31 18:37:22	31-Mar-2012
	2012-04-01 23:11:12	1-Apr-2012
	2012-04-02 11:26:38	2-Apr-2012
	2012-04-03 12:22:31	3-Apr-2012

```
SELECT
    order_date AS 'Order Date'
    , DATE_FORMAT(order_date, '%e-%b-%Y')
        AS 'Order Date Formatted'
FROM
    orders
```