The Syntax of a SELECT Query

```
SELECT [columns to return]
FROM [schema.table]
WHERE [conditional filter statements]
GROUP BY [columns to group on]
HAVING [conditional filter statements that are run after grouping]
ORDER BY [columns to sort on]
```

- `SELECT` : Choose columns to show. **Required**.
- `FROM` : Choose the table (and schema). **Required**.
- `WHERE` : Filter rows before grouping or selecting.
- `GROUP BY` : Group rows with the same values.
- `HAVING` : Filter groups after grouping.
- `ORDER BY` : Sort the data by columns.

Selecting Columns and Limiting the Number of Rows

- `FROM schema.table` specifies the schema and table name.

```
SELECT * FROM farmers_market.product
```

Table 2.1

product_id	product_name	product_size	product_category_id	product_qty_type
1	Habanero Peppers - Organic	medium	1	lbs
2	Jalapeno Peppers - Organic	small	1	lbs
3	Poblano Peppers - Organic	large	1	unit
4	Banana Peppers - Jar	8 oz	3	unit
5	Whole Wheat Bread	1.5 lbs	3	unit
6	Cut Zinnias Bouquet	medium	5	unit
7	Apple Pie	10"	3	unit
8	Cherry Pie	10"	3	unit
9	Sweet Potatoes	medium	1	lbs
10	Eggs	1 dozen	6	unit
11	Pork Chops	1 lb	6	lbs
12	Baby Salad Lettuce Mix - Bag	1/2 lb	1	unit
13	Baby Salad Lettuce Mix	1 lb	1	lbs
14	Red Potatoes	NULL	1	NULL
15	Red Potatoes - Small		1	NULL
16	Sweet Corn	Ear	1	unit
17	Carrots	sold by weight	1	lbs
18	Carrots - Organic	bunch	1	unit
19	Farmer's Market Resuable Shopping Bag	medium	7	unit
20	Homemade Beeswax Candles	6"	7	unit
21	Organic Cherry Tomatoes	pint	1	unit
22	Roma Tomatoes	medium	1	lbs
23	Maple Syrup - Jar	8 oz	2	unit

- Limit for the first five rows:

```
SELECT *
FROM farmers_market.product
LIMIT 5
```

Table 2.2

product_id	product_name	product_size	product_category_id	product_qty_type
1	Habanero Peppers - Organic	medium	1	lbs
2	Jalapeno Peppers - Organic	small	1	lbs
3	Poblano Peppers - Organic	large	1	unit
4	Banana Peppers - Jar	8 oz	3	unit
5	Whole Wheat Bread	1.5 lbs	3	unit

- Different database systems use different syntax to limit results:
  - MS SQL Server: Use `TOP` before `SELECT`.
  - Oracle: Use `WHERE ROWNUM <= number`.
  - MySQL: Use `LIMIT` at the end of the query.

- Use line breaks and indentation for readability. It does not affect execution.
- To specify columns, list column names after SELECT, separated by commas.

```sql
SELECT product_id, product_name
FROM farmers_market.product
LIMIT 5
```

Table 2.3

product_id	product_name
1	Habanero Peppers - Organic
2	Jalapeno Peppers - Organic
3	Poblano Peppers - Organic
4	Banana Peppers - Jar
5	Whole Wheat Bread

- List column names explicitly instead of using `*`:
  - `Schema Changes`: `SELECT *` can cause issues if the table changes.
  - `Consistent Output`: Listing columns ensures consistent output.
- Preventing Breakages:
  - `Automated Processes`: Unexpected changes can cause failures.
  - `Error Detection`: Listing columns alerts you if a column is removed or renamed.

The ORDER BY Clause: Sorting Results

- `ORDER BY` clause sorts rows by columns.
  - Sort order: `ASC` (ascending) or `DESC` (descending).
  - `ASC`: Text alphabetically, numbers low to high.
  - `DESC`: Reverse order.
  - MySQL: `NULL` values first in ascending order.
  - Default: Ascending.
- ASC:

```sql
SELECT product_id, product_name
FROM farmers_market.product
ORDER BY product_name
LIMIT 5
```

Table 2.4

product_id	product_name
7	Apple Pie
13	Baby Salad Lettuce Mix
12	Baby Salad Lettuce Mix - Bag
4	Banana Peppers - Jar
17	Carrots

- DESC:

Table 2.5

product_id	product_name
23	Maple Syrup - Jar
22	Roma Tomatoes
21	Organic Cherry Tomatoes
20	Homemade Beeswax Candles
19	Farmer's Market Resuable Shopping Bag

- Rows in Table 2.5 differ because `ORDER BY` runs before `LIMIT`.
  - `LIMIT` must be after `ORDER BY`.
- First, sort by `market_date`, then by `vendor_id`.

```
SELECT market_date, vendor_id, booth_number
FROM farmers_market.vendor_booth_assignments
ORDER BY market_date, vendor_id
LIMIT 5
```

Table 2.6

market_date	vendor_id	booth_number
2019-04-03	1	2
2019-04-03	3	1
2019-04-03	4	7
2019-04-03	7	11
2019-04-03	8	6

Simple Inline Calculations

```
SELECT
    market_date, customer_id,
    vendor_id, quantity,
    cost_to_customer_per_qty
FROM farmers_market.customer_purchases
LIMIT 5
```

Table 2.7

market_date	customer_id	vendor_id	quantity	cost_to_customer_per_qty
2019-07-03	14	7	0.99	6.99
2019-07-03	14	7	2.18	6.99
2019-07-03	15	7	1.53	6.99
2019-07-03	16	7	2.02	6.99
2019-07-03	22	7	0.66	6.99

- Multiply `quantity` and `cost_per_quantity`:

```sql
SELECT
    market_date, customer_id,
    vendor_id, quantity,
    cost_to_customer_per_qty,
    quantity * cost_to_customer_per_qty
FROM farmers_market.customer_purchases
LIMIT 5
```

Table 2.8

market_date	customer_id	vendor_id	quantity	cost_to_customer_per_qty	quantity * cost_to_customer_per_q
2019-07-03	14	7	0.99	6.99	6.9201
2019-07-03	14	7	2.18	6.99	15.2382
2019-07-03	15	7	1.53	6.99	10.6947
2019-07-03	16	7	2.02	6.99	14.1198
2019-07-03	22	7	0.66	6.99	4.6134

- Use `AS` to give a calculated column a meaningful name.
    - Example: Assign alias `price` to the result.

```sql
SELECT
    market_date, customer_id,
    vendor_id,
    quantity * cost_to_customer_per_qty AS price
FROM farmers_market.customer_purchases
LIMIT 5
```

Table 2.9

market_date	customer_id	vendor_id	price
2019-07-03	14	7	6.9201
2019-07-03	14	7	15.2382
2019-07-03	15	7	10.6947
2019-07-03	16	7	14.1198
2019-07-03	22	7	4.6134

- `AS` is optional in MySQL.
- The query will return the same result without it.

```sql
SELECT
    market_date,
    customer_id,
    vendor_id,
    quantity * cost_to_customer_per_qty price
FROM farmers_market.customer_purchases
LIMIT 5
```

- Aggregating Data:
    - Summarize data across multiple rows.
    - Examples: `SUM`, `AVG`, `COUNT`.
    - Covered in Chapter 6.
- Current calculations apply to each row individually.

    - Example: Calculate price by multiplying quantity by cost per unit.
    - These do not summarize or combine data across rows.

More Inline Calculation

- A `SQL` function takes inputs, performs an operation, and returns a value.
- Use functions to modify raw values before displaying them.
- Syntax for calling a `SQL` function:
    - Input parameters can be a column name or a constant value.
    - Refer to documentation for correct parameters: MySQL Documentation.
- Example: `ROUND` function rounds values in the `price` column to two decimal places.

```
SELECT
    market_date, customer_id,
    vendor_id,
    ROUND(quantity * cost_to_customer_per_qty, 2) AS price
FROM farmers_market.customer_purchases
LIMIT 5
```

Table 2.10

market_date	customer_id	vendor_id	price
2019-07-03	14	7	6.92
2019-07-03	14	7	15.24
2019-07-03	15	7	10.69
2019-07-03	16	7	14.12
2019-07-03	22	7	4.61

- `ROUND` can accept negative values for the second parameter to round left of the decimal point.
    - Example: `ROUND(1245, -2)` returns 1200.
- `SQL` functions can manipulate text data.
- `CONCAT` function joins strings together.
- In the `customer` table, `first_name` and `last_name` are in separate columns.

```
SELECT *
FROM farmers_market.customer
LIMIT 5
```

Table 2.11

customer_id	customer_first_name	customer_last_name	customer_zip
1	Jane	Connor	22801
2	Manuel	Diaz	22821
3	Bob	Wilson	22821
4	Deanna	Washington	22801
5	Abigail	Harris	22801

- Use `CONCAT` to combine `first_name`, a space, and `last_name` into `customer_name`.

```
SELECT
    customer_id,
    CONCAT(customer_first_name, " ", customer_last_name) AS customer_name
FROM farmers_market.customer
LIMIT 5
```

Table 2.12

customer_id	customer_name
1	Jane Connor
2	Manuel Diaz
3	Bob Wilson
4	Deanna Washington
5	Abigail Harris

- First, sort by `last_name` and `first name`.
- Then concatenate `first_name` and `last_name`.

```
SELECT
    customer_id,
    CONCAT(customer_first_name, " ", customer_last_name) AS customer_name
FROM farmers_market.customer
ORDER BY customer_last_name, customer_first_name
LIMIT 5
```

Table 2.13

customer_id	customer_name
7	Jessica Armenta
6	Betty Bullard
1	Jane Connor
17	Carlos Diaz
2	Manuel Diaz

- You can nest functions.
    - Example: Use `UPPER` to convert the concatenated name to uppercase.

```
SELECT
    customer_id,
    UPPER(CONCAT(customer_last_name, ", ", customer_first_name)) AS customer_name
FROM farmers_market.customer
ORDER BY customer_last_name, customer_first_name
LIMIT 5
```

Table 2.14

customer_id	customer_name
7	ARMENTA, JESSICA
6	BULLARD, BETTY
1	CONNOR, JANE
17	DIAZ, CARLOS
2	DIAZ, MANUEL

- Notes:
    - Sorting is done on `customer_last_name` and `customer_first_name`, not on `customer_name`.
    - Aliases might not be reusable in other parts of the query.
    - You can use functions in `ORDER BY`. Example: `ORDER BY UPPER(customer_last_name)` to sort uppercased last names.

Evaluating Query Output

- Steps to ensure SQL query results are as expected:
    - Run the query with `LIMIT` to preview the first few rows.
        - Verify changes in the output.
        - Check column names and values.
    - Verify Total Rows:
        - Run without `LIMIT` or use `COUNT` to confirm total rows.
- Use the Query Editor to review results:
    - Quick sanity check, not a substitute for full quality control.
    - Remove `LIMIT` to review the full dataset.
    - In MySQL Workbench, use "Don't Limit" under the Query menu.
- Run the query to generate full output:
    - Check total row count to match expectations.
    - Example: 21 rows in `customer_purchases`.
    - In MySQL Workbench, check the Message in the Output section.
- Review the resulting dataset ("Result Grid" in MySQL Workbench):
    - Check column headers.
    - Spot-check values.
    - Verify sorting if using `ORDER BY`.
- Manually sort each column:
    - Example: Sort by `market_date` in ascending order.
    - Sort by `vendor_id` column.
- Check minimum and maximum values in each column:
    - Find errors like unexpected negative values or NULLs.
    - Look for strings starting with numbers or spaces, or other anomalies.

Exercises Using the Included Database

- Exercises for the `customer` table:
    - Columns and example rows are shown in Figure 2.11.

1. Retrieve all columns from the `customer` table.
2. Display all columns and 10 rows, sorted by `last_name`, then `first_name`.
3. List all `customer_id` and `first_name`, sorted by `first_name`.