WeCloudData

# SQL Fundamentals

Kick off your career in data science & analytics

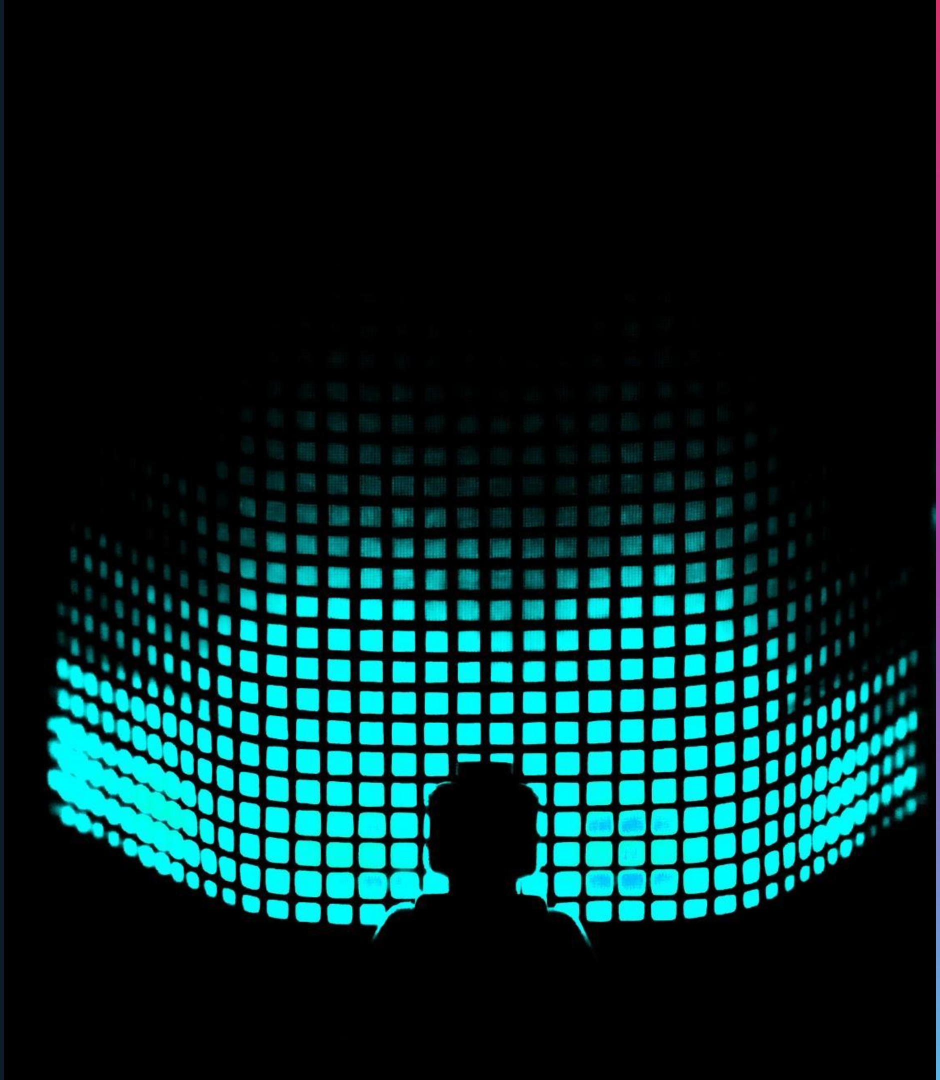
# Module 3:

## Basic Queries

# Learning Objectives

In this module, we will continue to learn how to write queries in SQL. Specifically, we will share with you:

- SQL terminology
- SQL queries



**SQL Terminology**

SQL Queries

# Agenda.

# SQL Terminology

## Basic Queries

Here are some basic SQL **terms** to get you started:

- Syntax
- Statement
- Clause
- Operator
- Expression

# SQL Terms: Syntax

## Basic Queries

SQL is followed by a unique set of rules and guidelines called **syntax**.

All SQL **statements** start with any of the following keywords:
**SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW**

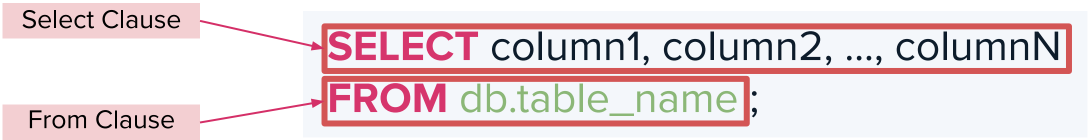
*Note:*

- All the statements end with a semicolon (**;**)
- SQL syntax is case insensitive, which means ***SELECT*** and ***select*** have same meaning in SQL statements
  - [Exception] Table names in MySQL are case sensitive

# SQL Terms: Statements, Clauses

## Basic Queries

SQL **statements** are used to perform tasks such as update data on a database or retrieve data from a database.



Select Clause

From Clause

```
SELECT column1, column2, ..., columnN
FROM db.table_name;
```

```
CREATE TABLE table_name [AS] (
        column1 datatype [options],
        column2 datatype [options],
        ...,
        columnN datatype [options],
        PRIMARY KEY (column1, column5)
);
```

# SQL Terms: Operators

## Basic Queries

An **operator** is a reserved word or character that is primarily used in an SQL statement **WHERE** clause.

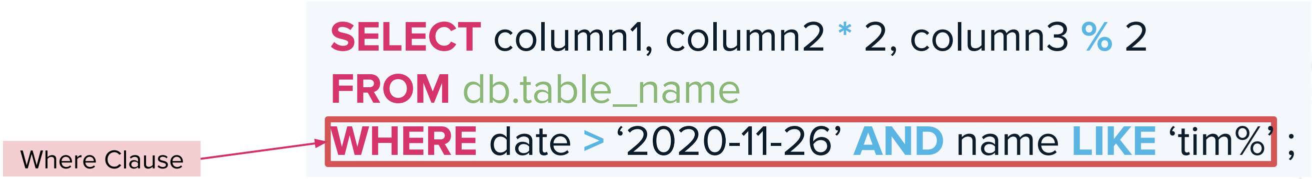
This allows it to perform operation(s) such as comparisons and arithmetic operations.

```
SELECT column1, column2 * 2, column3 % 2
FROM db.table_name
WHERE date > ‘2020-11-26’ AND name LIKE ‘tim%’ ;
```

Where Clause

# SQL Terms: Operators (Cont’d)

## Basic Queries



```
SELECT column1, column2 * 2, column3 % 2
FROM db.table_name
WHERE date > ‘2020-11-26’ AND name LIKE ‘tim%’ ;
```

Where Clause

*Operators used here:*

| | | |
|---|---|---|
| *, % | ➜ | Arithmetic Operator |
| > | ➜ | Comparison Operator |
| LIKE, AND | ➜ | Logic Operator |

SQL Terminology

**SQL Queries**

# Agenda.

# SQL Terminology

## Basic Queries

Here are some basic SQL **queries** to get you started:

- Select rows
- Select columns
- Unique values
- Sorting results
- Counts

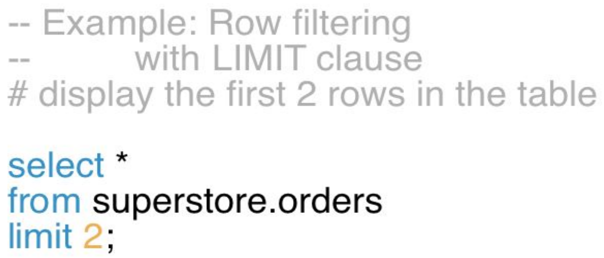
# SQL Terminology

## Basic Queries

| Order | Clause | Description |
|---|---|---|
| 1 | FROM | Identifies which table(s) are being queried |
| 2 | WHERE | Filter records that satisfy a specified condition |
| 3 | GROUP BY | Used to group rows that have the same values |
| 4 | HAVING | Used in combination with the GROUP BY clause to return only groups of rows whose aggregated values meet the specified conditions (i.e. filters aggregated data) |
| 5 | SELECT | Specifies columns from which data values are to be retrieved from |
| 6 | ORDER BY | Sorts the data |
| 7 | LIMIT | Limits the data to a row count |

# Select Rows from a Table: Limit

## Basic Queries



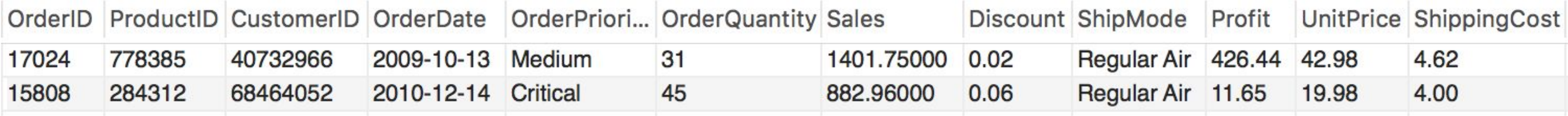
```
-- Example: Row filtering
--          with LIMIT clause
# display the first 2 rows in the table

select *
from superstore.orders
limit 2;
```

### *Syntax*

```
SELECT *
FROM db.table_name
LIMIT 5;
```

*NOTE*:

- Typically we are dealing with large tables that have thousands, or even millions of rows
- Be cautious when you run **SELECT** without the **LIMIT** (as it will get all rows)



| OrderID | ProductID | CustomerID | OrderDate | OrderPriori... | OrderQuantity | Sales | Discount | ShipMode | Profit | UnitPrice | ShippingCost |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 17024 | 778385 | 40732966 | 2009-10-13 | Medium | 31 | 1401.75000 | 0.02 | Regular Air | 426.44 | 42.98 | 4.62 |
| 15808 | 284312 | 68464052 | 2010-12-14 | Critical | 45 | 882.96000 | 0.06 | Regular Air | 11.65 | 19.98 | 4.00 |

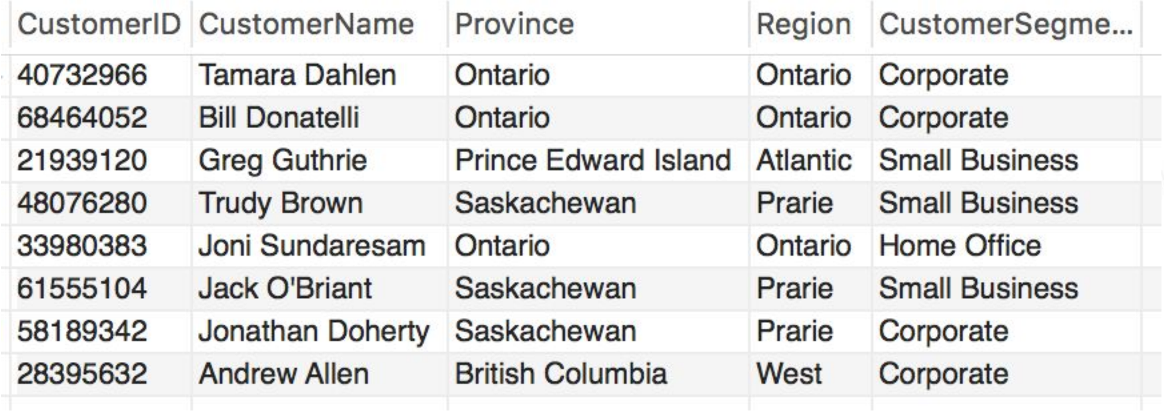
# Select ALL Columns from a Table: Using *

## Basic Queries

```
-- select all the columns
select *
from superstore.customer;
```

### *Syntax*

```
SELECT *
FROM db.table_name;
```



| CustomerID | CustomerName | Province | Region | CustomerSegme... |
|---|---|---|---|---|
| 40732966 | Tamara Dahlen | Ontario | Ontario | Corporate |
| 68464052 | Bill Donatelli | Ontario | Ontario | Corporate |
| 21939120 | Greg Guthrie | Prince Edward Island | Atlantic | Small Business |
| 48076280 | Trudy Brown | Saskachewan | Prarie | Small Business |
| 33980383 | Joni Sundaresam | Ontario | Ontario | Home Office |
| 61555104 | Jack O'Briant | Saskachewan | Prarie | Small Business |
| 58189342 | Jonathan Doherty | Saskachewan | Prarie | Corporate |
| 28395632 | Andrew Allen | British Columbia | West | Corporate |

# Select Rows from a Table: Where

## Basic Queries

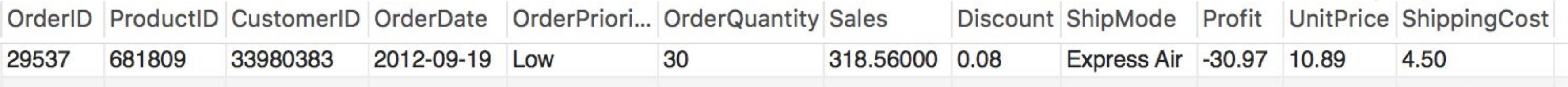
```
-- Example: Row filtering
-- with WHERE clause
select *
from superstore.orders
where OrderDate > '2011-12-01';
```

### *Syntax*

```
SELECT *
FROM db.table_name
WHERE [condition];
```

*NOTE*:

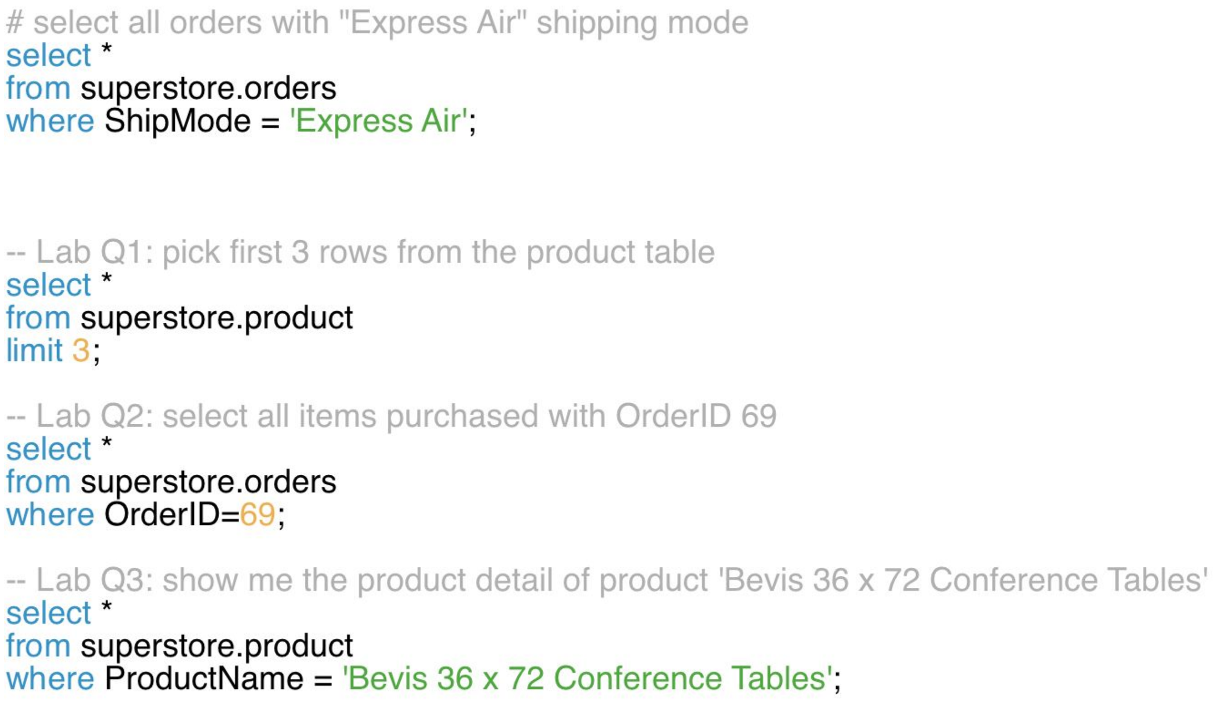
- WHERE is used for filtering data when you know the condition
- It is one of the most commonly used clause in SQL



| OrderID | ProductID | CustomerID | OrderDate | OrderPriori... | OrderQuantity | Sales | Discount | ShipMode | Profit | UnitPrice | ShippingCost |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 29537 | 681809 | 33980383 | 2012-09-19 | Low | 30 | 318.56000 | 0.08 | Express Air | -30.97 | 10.89 | 4.50 |

# Select Rows from a Table: More WHERE Examples

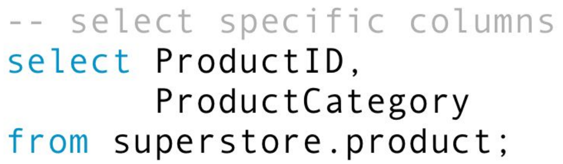
## Basic Queries



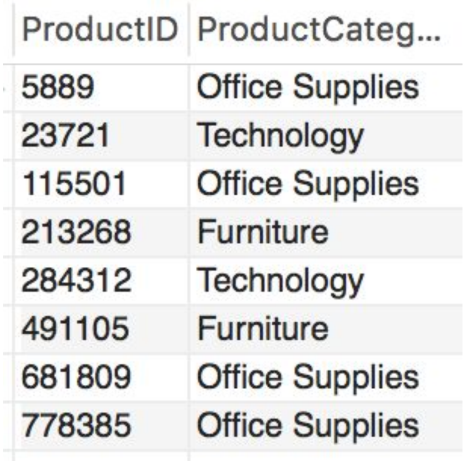
```
# select all orders with "Express Air" shipping mode
select *
from superstore.orders
where ShipMode = 'Express Air';



-- Lab Q1: pick first 3 rows from the product table
select *
from superstore.product
limit 3;

-- Lab Q2: select all items purchased with OrderID 69
select *
from superstore.orders
where OrderID=69;

-- Lab Q3: show me the product detail of product 'Bevis 36 x 72 Conference Tables'
select *
from superstore.product
where ProductName = 'Bevis 36 x 72 Conference Tables';
```

# Select Columns from a Table: Projection

## Basic Queries



```
-- select specific columns
select ProductID,
       ProductCategory
from superstore.product;
```



| ProductID | ProductCateg... |
|---|---|
| 5889 | Office Supplies |
| 23721 | Technology |
| 115501 | Office Supplies |
| 213268 | Furniture |
| 284312 | Technology |
| 491105 | Furniture |
| 681809 | Office Supplies |
| 778385 | Office Supplies |

### *Syntax*

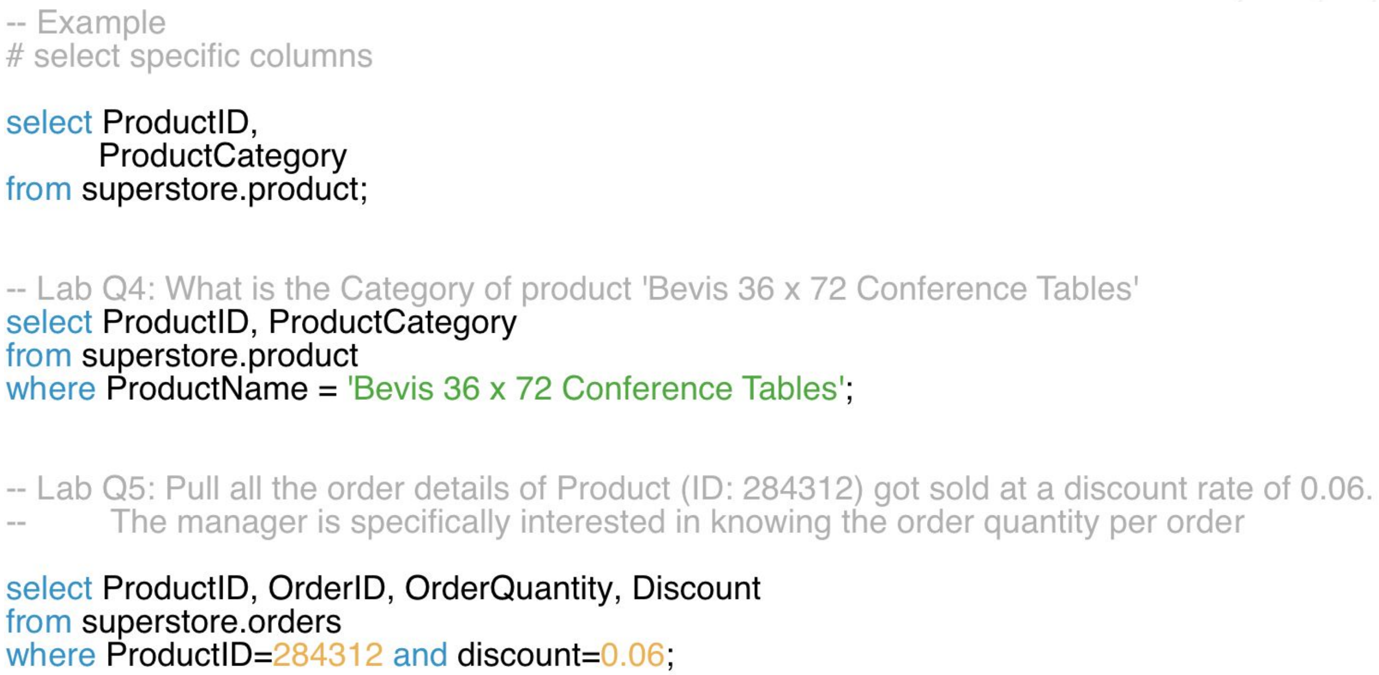
```
SELECT column1, ..., columnN
FROM db.table_name;
```

*NOTE*:

- Column projection will make your queries more efficient because your queries will be dealing with less data
- In a real workplace, you may need to deal with tables that have hundreds and thousands of columns
- A well maintained data dictionary is very helpful

# Select Columns from a Table: More PROJECTION Examples

## Basic Queries

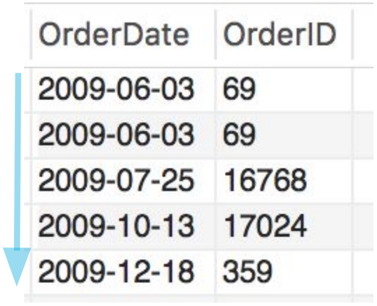


```
-- Example
# select specific columns

select ProductID,
       ProductCategory
from superstore.product;


-- Lab Q4: What is the Category of product 'Bevis 36 x 72 Conference Tables'
select ProductID, ProductCategory
from superstore.product
where ProductName = 'Bevis 36 x 72 Conference Tables';


-- Lab Q5: Pull all the order details of Product (ID: 284312) got sold at a discount rate of 0.06.
--        The manager is specifically interested in knowing the order quantity per order

select ProductID, OrderID, OrderQuantity, Discount
from superstore.orders
where ProductID=284312 and discount=0.06;
```

# Sorting Results: Order By

## Basic Queries

```sql
-- Example: What is the earliest
-- order date in the Orders table
select OrderDate, OrderID
from superstore.orders
order by OrderDate asc
limit 5;
```



| OrderDate | OrderID |
|---|---|
| 2009-06-03 | 69 |
| 2009-06-03 | 69 |
| 2009-07-25 | 16768 |
| 2009-10-13 | 17024 |
| 2009-12-18 | 359 |

### *Syntax*

```
SELECT column1, column5
FROM db.table_name
ORDER BY column1, column5 [ASC | DESC];
```

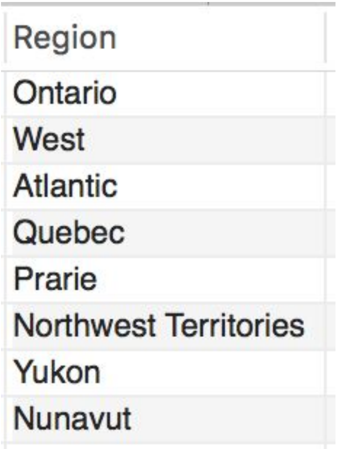
*NOTE*:

- Sorting may take some time with large tables

# Select Unique Values from a Column: Distinct

## Basic Queries

```
-- Example: what regions do customers come from?
-- Select unique regions from customer table
select distinct Region
from superstore.customer;
```



| Region |
|---|
| Ontario |
| West |
| Atlantic |
| Quebec |
| Prarie |
| Northwest Territories |
| Yukon |
| Nunavut |

### *Syntax*

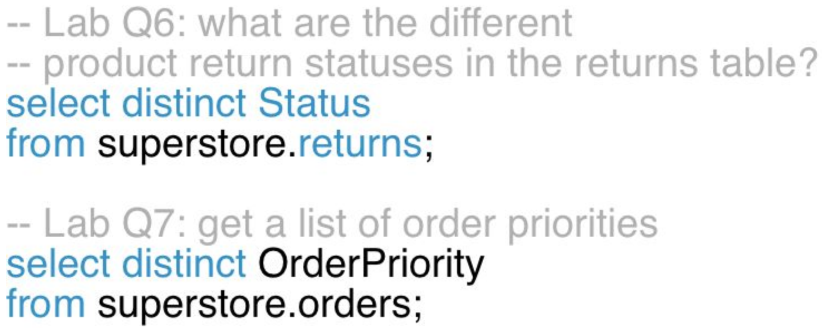
```
SELECT DISTINCT columnK
FROM db.table_name;
```

*NOTE*:

- It’s always important to understand the unique number of product categories, user segments
- You may find out that you have 6 unique values while you only have 5 categories
  - That might be because of incorrect data input or even missing data
  - In that case, you need to handle the data with care

# Select Unique Values from a Column: More DISTINCT Examples

## Basic Queries



```
-- Lab Q6: what are the different
-- product return statuses in the returns table?
select distinct Status
from superstore.returns;

-- Lab Q7: get a list of order priorities
select distinct OrderPriority
from superstore.orders;
```

# Counting Results: Count

## Basic Queries

```
-- Example: How many customers
-- have made purchases in the
-- past?
select count(*)
from superstore.customer;


-- Example: How many provinces
-- have our products been sold to?
select count(distinct Province)
from superstore.customer;
```

### *Syntax*

```
SELECT COUNT(*)
FROM db.table_name
WHERE [condition];

SELECT COUNT(DISTINCT columnK)
FROM db.table_name
WHERE [condition];
```

*NOTE*:

- One thing you should always do is understand the number of records for some dimension
  - E.g. Total number of rows in your table, total number of customers, etc.
- You can combine **count** with **distinct**. This is useful when you have users who purchase multiple items at each transaction.