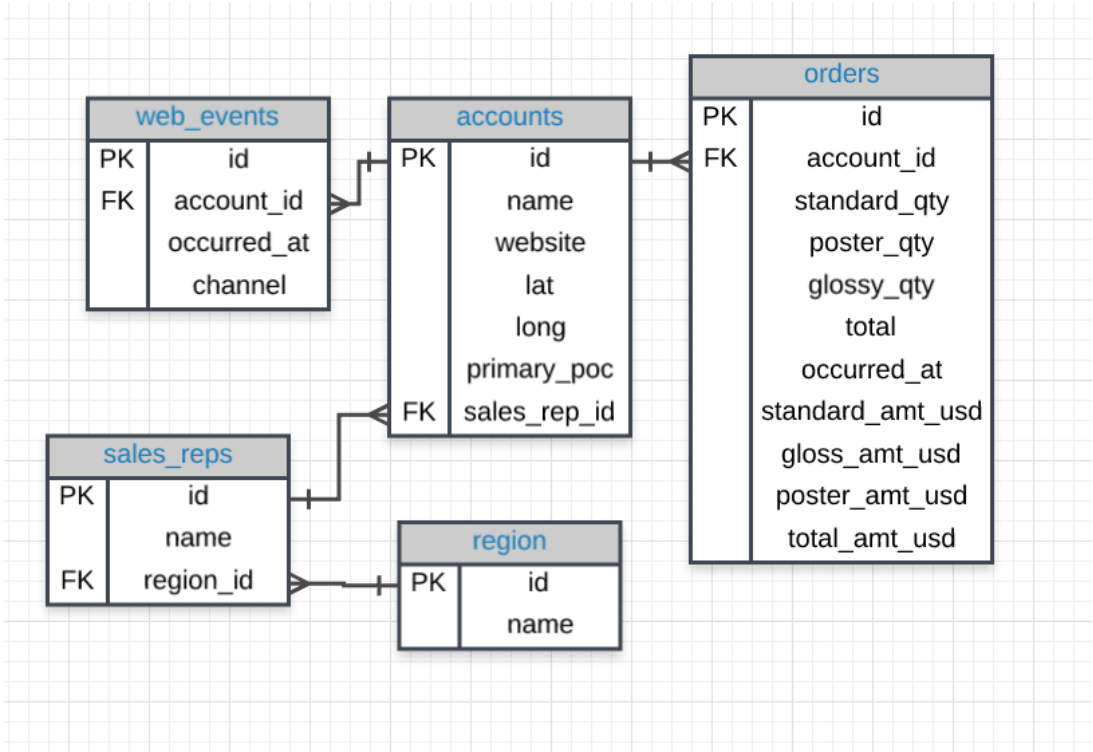
SQL

AMIT

Entity Relationship Diagrams



Entity Relationship Diagrams

An **entity-relationship diagram** (ERD) is a common way to view data in a database.
Below is the ERD for the database we will use from Parch & Posey.
These diagrams help you visualize the data you are analyzing including:

1. The names of the tables.
2. The columns in each table.
3. The way the tables work together.

Select

The SELECT statement is used to select data from a database.

Example :

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

Where

The WHERE clause is used to filter records.
It is used to extract only those records that fulfill a specified condition.

Example :

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

AMIT

AND, OR and NOT Operators

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.
- The NOT operator displays a record if the condition(s) is NOT TRUE.

Example AND:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND
condition3 ...;
-----------------------------------
SELECT * FROM Customers
WHERE Country='Germany' AND City='
Berlin';
```

AMIT

Example OR :

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR
condition3 ...;
```

```
SELECT * FROM Customers
WHERE City='Berlin' OR City='Münch
en';
```

Example NOT :

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

```
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

AMIT

Examples:

```
SELECT * FROM Customers
WHERE Country='Germany' AND (City=
'Berlin' OR City='München');
```

```
SELECT * FROM Customers
WHERE NOT Country='Germany’
AND Not Country='USA';
```

AMIT

ORDER BY :

The ORDER BY keyword is used to sort the result-set in ascending or descending order.
The ORDER BY keyword sorts the records in ascending order by default.
To sort the records in descending order, use the DESC keyword.

Example :

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2,
... ASC|DESC;
```

AMIT

Examples:

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

```
SELECT * FROM Customers
ORDER BY Country ASC,
CustomerName DESC;
```

AMIT

INSERT INTO :

The INSERT INTO statement is used to insert new records in a table.

Example :

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO Customers (CustomerName, ContactName, Address,
City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen
21', 'Stavanger', '4006', 'Norway');
```

AMIT

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.
We will have to use the IS NULL and IS NOT NULL operators instead.

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

```
SELECT CustomerName, ContactName,
Address
FROM Customers
WHERE Address IS NULL;
```

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

```
SELECT CustomerName, ContactName,
Address
FROM Customers
WHERE Address IS NOT NULL;
```

AMIT

UPDATE :

The UPDATE statement is used to modify the existing records in a table.

Example :

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

AMIT

```sql
UPDATE Customers
SET ContactName='Juan';

UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

DELETE:

The DELETE statement is used to delete existing records in a table.

Example :

```
DELETE FROM table_name WHERE condition;
```

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

AMIT

SELECT TOP :

The SELECT TOP clause is used to specify the number of records to return.

Example :

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

```
SELECT * FROM Customers
LIMIT 3;
```

AMIT

MIN() and MAX() Functions :

The MIN() function returns the smallest value of the selected column.
The MAX() function returns the largest value of the selected column.

Example :

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

AMIT

```sql
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

```sql
SELECT MAX(Price) AS LargestPrice
FROM Products;
```

COUNT(), AVG() and SUM() Functions:

The COUNT() function returns the number of rows that matches a specified criterion.

Example :

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

AMIT

LIKE Operator:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

Example :

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

AMIT

The following SQL statement selects all customers with a CustomerName starting with "a":

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

The following SQL statement selects all customers with a CustomerName ending with "a":

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%a';
```

The following SQL statement selects all customers with a CustomerName that have "or" in any position:

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%or%';
```

IN Operator :

The IN operator allows you to specify multiple values in a WHERE clause.

Example :

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

AMIT

```sql
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');

SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

BETWEEN Operator:

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

Example :

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

AMIT

```sql
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

```sql
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3)
```

```sql
SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon
Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

```sql
SELECT * FROM Products
WHERE ProductName BETWEEN "Carnarvon Tigers" AND "Chef Anton's
Cajun Seasoning"
ORDER BY ProductName;
```

```sql
SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella
di Giovanni'
ORDER BY ProductName;
```

```sql
SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
```

Joins :

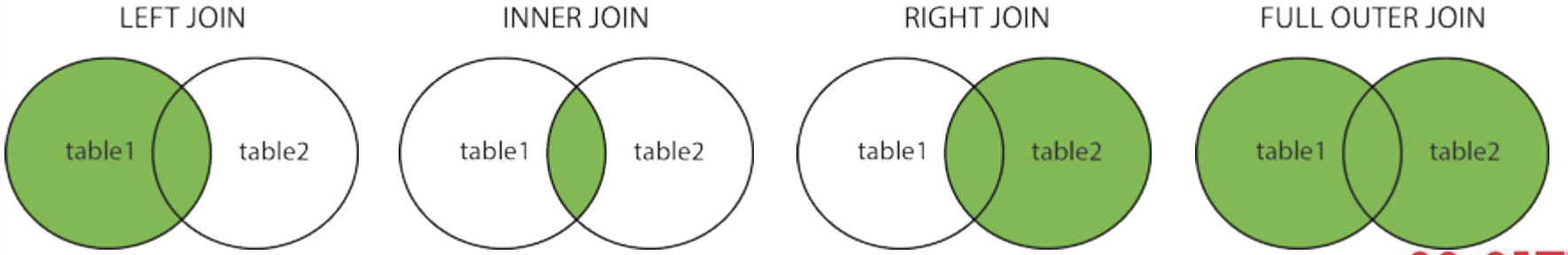
A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

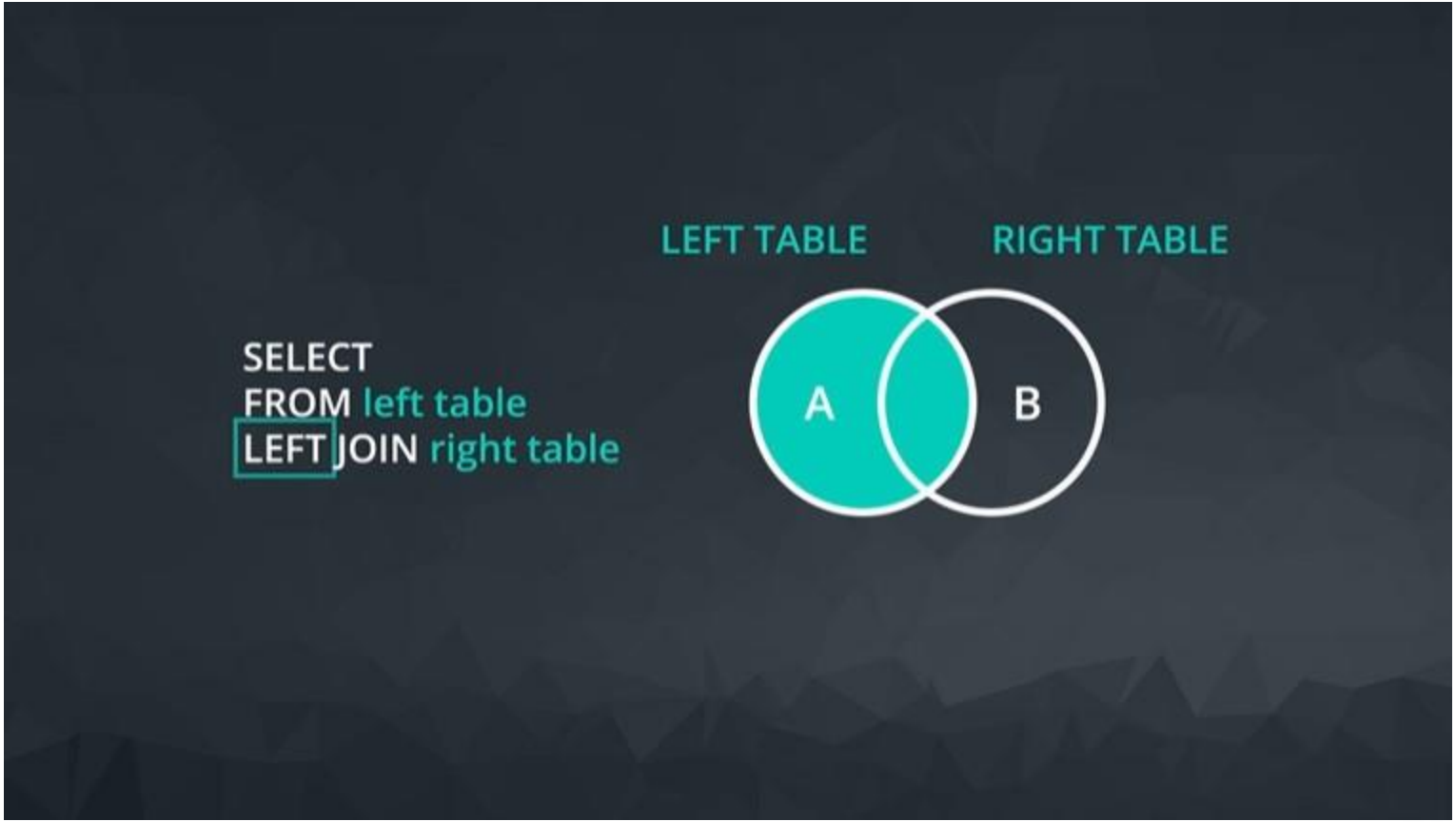
Example :

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

AMIT

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table




LEFT TABLE
RIGHT TABLE
SELECT
FROM left table
LEFT JOIN right table
A
B

ORDERS ACCOUNTS


```
SELECT a.id, a.name, o.total
FROM orders o
LEFT JOIN accounts a
ON o.account_id = a.id
```

ORDERS

id	account_id	total
1	1001	169
2	1001	288
17	1011	541
18	1021	539

ACCOUNTS

id	name
1001	Walmart
1011	Exxon Mobil
1021	Apple
1031	Berkshire Hathaway
1041	McKesson
1051	UnitedHealth Group
1061	CVS Health

AMIT

ORDERS ACCOUNTS


```
SELECT a.id, a.name, o.total
FROM orders o
RIGHT JOIN accounts a
ON o.account_id = a.id
```

ORDERS

id	account_id	total
1	1001	169
2	1001	288
17	1011	541
18	1021	539

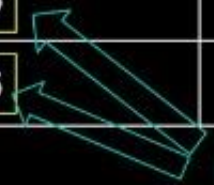
ACCOUNTS

id	name
1001	Walmart
1011	Exxon Mobil
1021	Apple
1031	Berkshire Hathaway
1041	McKesson
1051	UnitedHealth Group
1061	CVS Health

AMIT

FROM Country

countryid	countryName
1	India
2	Nepal
3	United States
4	Canada
5	Sri Lanka
6	Brazil



Included In LEFT JOIN

LEFT JOIN State

stateid	countryid	stateName
1	1	Maharashtra
2	1	Punjab
3	2	Kathmandu
4	3	California
5	3	Texas
6	4	Alberta

AMIT

The resulting table will look like:

countryid	countryName	stateName
1	India	Maharashtra
1	India	Punjab
2	Nepal	Kathmandu
3	United States	California
3	United States	Texas
4	Canada	Alberta
5	Sri Lanka	NULL
6	Brazil	NULL

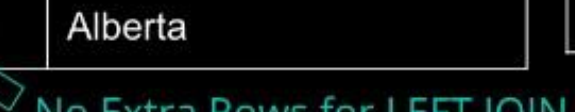
AMIT

FROM State

stateid	countryid	stateName
1	1	Maharashtra
2	1	Punjab
3	2	Kathmandu
4	3	California
5	3	Texas
6	4	Alberta

LEFT JOIN Country

countryid	countryName
1	India
2	Nepal
3	United States
4	Canada
5	Sri Lanka
6	Brazil



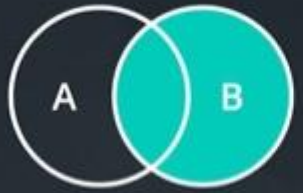
AMIT

The resulting table will look like:

countryid	countryName	stateName
1	India	Maharashtra
1	India	Punjab
2	Nepal	Kathmandu
3	United States	California
3	United States	Texas
4	Canada	Alberta

AMIT

ORDERS ACCOUNTS




```
SELECT a.id, a.name, o.total
FROM orders o
RIGHT JOIN accounts a
ON o.account_id = a.id
```

NOT MATCHED →

RIGHT JOIN RESULT

id	account_id	total
1001	Walmart	169
1001	Walmart	288
1011	Exxon Mobil	541
1021	Apple	539
1031	Berkshire Hathaway	
1041	McKesson	
1051	UnitedHealth Group	
1061	CVS Health	

AMIT

ACCOUNTS

id	name
1001	Walmart
1011	Exxon Mobil
1021	Apple
1031	Berkshire Hathaway
1041	McKesson
1051	UnitedHealth Group
1061	CVS Health

```
SELECT a.id, a.name, o.total
FROM accounts a
LEFT JOIN orders o
ON o.account_id = a.id
```

ORDERS

id	account_id	total
1	1001	169
2	1001	288
17	1011	541
18	1021	539

AMIT

THESE GIVE THE SAME RESULTING TABLE

```
SELECT a.id, a.name, o.total
FROM orders o
RIGHT JOIN accounts a
ON o.account_id = a.id
```

```
SELECT a.id, a.name, o.total
FROM accounts a
LEFT JOIN orders o
ON o.account_id = a.id
```

RESULT

id	account_id	total
1001	Walmart	169
1001	Walmart	288
1011	Exxon Mobil	541
1021	Apple	539
1031	Berkshire Hathaway	
1041	McKesson	
1051	UnitedHealth Group	
1061	CVS Health	

AMIT

INNER JOIN

Country

countryid	countryName
1	India
2	Nepal
3	United States
4	Canada
5	Sri Lanka
6	Brazil

State

stateid	countryid	stateName
1	1	Maharashtra
2	1	Punjab
3	2	Kathmandu
4	3	California
5	3	Texas
6	4	Alberta

AMIT

LEFT JOIN :

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

Example :

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

AMIT

```
SELECT a.id, a.name, o.total
FROM orders o
LEFT JOIN accounts a
ON o.account_id = a.id
```

```
SELECT c.countryid, c.countryName, s.stateName
FROM Country c
LEFT JOIN State s
ON c.countryid = s.countryid;
```

AMIT

```
SELECT orders.*, accounts.*
FROM orders
LEFT JOIN accounts
ON orders.account_id = accounts.id
WHERE accounts.sales_rep_id = 321500
```

INNER JOIN :

The INNER JOIN keyword selects records that have matching values in both tables.

Example :

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

AMIT

```
SELECT orders.* FROM orders JOIN accounts ON
orders.account_id = accounts.id;
```

```
SELECT accounts.name, orders.occurred_at FROM orders JOIN
accounts ON orders.account_id = accounts.id;
```

```
SELECT * FROM orders JOIN accounts ON orders.account_id =
accounts.id;
```

```
SELECT orders.* FROM orders JOIN accounts ON
orders.account_id = accounts.id;
```

AMIT

```sql
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID = Customers.CustomerID;
```

RIGHT JOIN:

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1).
The result is 0 records from the left side, if there is no match.

Example :

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

AMIT

```
SELECT a.id, a.name, o.total
FROM orders o
JOIN accounts a
ON o.account_id = a.id
```

AMIT

```sql
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees
ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

OUTER JOIN:

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Tip: FULL OUTER JOIN and FULL JOIN are the same.

Example :

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

AMIT

GROUP BY:

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

Example :

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

AMIT

```sql
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

```sql
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

```sql
SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders
 FROM Orders
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID
GROUP BY ShipperName;
```

AMIT

HAVING Clause:

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

Example :

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

AMIT

```
SELECT s.id, s.name, COUNT(*) num_accounts
FROM accounts a
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.id, s.name
HAVING COUNT(*) > 5
ORDER BY num_accounts;
```

```
SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders
FROM Orders
INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
WHERE LastName = 'Davolio' OR LastName = 'Fuller'
GROUP BY LastName
HAVING COUNT(Orders.OrderID) > 25;
```

AMIT

Case:

```
SELECT CASE WHEN total >= 2000
THEN 'At Least 2000' WHEN total >= 1000 AND total < 2000
THEN 'Between 1000 and 2000' ELSE 'Less than 1000’
END AS order_category,
COUNT(*) AS order_count
FROM orders
GROUP BY 1;
```

AMIT

```
SELECT id, account_id, occurred_at, channel,
CASE WHEN channel = 'facebook' THEN 'yes’
END AS is_facebook FROM web_events
ORDER BY occurred_at
```

```
SELECT id, account_id, occurred_at, channel,
 CASE WHEN channel = 'facebook' THEN 'yes' ELSE 'no’
END AS is_facebook FROM web_events
ORDER BY occurred_at
```

```
SELECT id, account_id, occurred_at, channel,
CASE WHEN channel = 'facebook' OR channel = 'direct' THEN 'yes' ELSE 'no’
END AS is_facebook FROM web_events ORDER BY occurred_at
```

AMIT

```
SELECT account_id, occurred_at, total, CASE WHEN total > 500 THEN 'Over 500’
WHEN total > 300 THEN '301 - 500' WHEN total > 100
THEN '101 - 300' ELSE '100 or under’
 END AS total_group FROM orders
```

```
SELECT account_id, occurred_at, total, CASE WHEN total > 500 THEN 'Over 500’
WHEN total > 300 AND total <= 500 THEN '301 - 500’
WHEN total > 100 AND total <=300 THEN '101 - 300' ELSE '100 or under’
END AS total_group FROM orders
```

AMIT

```
SELECT a.name, SUM(total_amt_usd) total_spent,
CASE WHEN SUM(total_amt_usd) > 200000 THEN 'top’
WHEN SUM(total_amt_usd) > 100000
THEN 'middle' ELSE 'low' END AS customer_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
GROUP BY a.name
ORDER BY 2 DESC;
```

```
SELECT CASE WHEN total > 500 THEN 'OVer 500' ELSE '500 or under’
END AS total_group, COUNT(*) AS order_count FROM orders GROUP BY 1
```

```
SELECT a.name, SUM(total_amt_usd) total_spent,
CASE WHEN SUM(total_amt_usd) > 200000 THEN 'top'
 WHEN SUM(total_amt_usd) > 100000 THEN 'middle' ELSE 'low'
END AS customer_level
FROM orders o JOIN accounts a ON o.account_id = a.id
WHERE occurred_at > '2015-12-31'
GROUP BY 1
ORDER BY 2 DESC;
```

```
SELECT COUNT(1) AS orders_over_500_units
FROM orders WHERE total > 500
```

AMIT

comment:

```
--SELECT * FROM Customers;



/*SELECT * FROM Customers;
SELECT * FROM Products;
SELECT * FROM Orders;
SELECT * FROM Categories;*/
SELECT * FROM Suppliers;
```

AMIT

Save Query to CSV

1- Write your query
2-Run it from run all
3-Save Export to csv

Then you have a table csv file you can use it in EXCEL OR POWER BI

AMIT

THANK YOU!