# 1. INNER JOIN

|  |
| --- |
| INNER JOIN Syntax  SELECT *column\_name(s)* FROM *table1* INNER JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;  SELECT \*  FROM table1 INNER JOIN table2 ON  table1.primaryKey=table2.table1Id INNER JOIN  table3 ON table1.primaryKey=table3.table1Id |



=> The point is: 2 table need common column => In order to join desired column

|  |
| --- |
| Query:  SELECT Orders.OrderID, Customers.CustomerName FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;  Explain:  Original: Orders Same column: Customer ID  Purpose: Join 2 column: OrderID and CustomerName |

## Advance Join 3 Table

|  |
| --- |
| SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID;  Explain:  Original: Orders Same column: Customer ID(Orders-Customers), ShipperID(Orders-Shippers)  Purpose: Join 3 column: OrderID and CustomerName and ShipperName |

## Advance combine AND Comparison

|  |
| --- |
| **First one: [without WHERE clause, just AND with ON]**  SELECT related\_tabid AS tabid, label, t.name  FROM relatedlists r  INNER JOIN tab t  ON t.tabid = r.tabid  AND t.name = 'Leads'  AND r.is\_active=1 and r.related\_tabid <> 0  AND t.is\_active=1  ORDER BY label  **Second one: [using WHERE clause, AND associated with where instead of ON ]**  SELECT related\_tabid AS tabid, label, t.name  FROM relatedlists r  INNER JOIN tab t  ON t.tabid = r.tabid  WHERE t.name = 'Leads'  AND r.is\_active=1 and r.related\_tabid <> 0  AND t.is\_active=1  ORDER BY label |

# 2. INSERT INTO SELECT

|  |
| --- |
| INSERT INTO SELECT Syntax  The INSERT INTO SELECT statement copies data from one table and inserts it into another table.   * INSERT INTO SELECT requires that data types in source and target tables match * The existing records in the target table are unaffected   Copy all columns from one table to another table:  INSERT INTO *table2* SELECT \* FROM *table1* WHERE *condition*;  Copy only some columns from one table into another table:  INSERT INTO *table2*(*column1*, *column2*, *column3*, ...) SELECT *column1*, *column2*, *column3*, ... FROM *table1* WHERE *condition*; => Condition from table where we select: table1 |

|  |
| --- |
| Example:  MariaDB [test2]> select \* from test2.customers;  +--------------+------+---------+  | CustomerName | City | Country |  +--------------+------+---------+  | A | HN | Germany |  | B | HCM | NewYork |  +--------------+------+---------+  MariaDB [test2]> select \* from test2.suppliers;  +--------------+------+---------+  | CustomerName | City | Country |  +--------------+------+---------+  | C | NTN | TEST |  | D | HKH | TEST2 |  +--------------+------+---------+  Query:  MariaDB [test2]> **INSERT INTO** Customers (CustomerName, City, Country) **SELECT** CustomerName, City, Country **FROM** Suppliers **WHERE** Country='TEST';  Query OK, 1 row affected (0.04 sec) |

# 3. REPLACE INTO

|  |
| --- |
| REPLACE works exactly like INSERT, except that if an old row in the table has the same value as a new row for a PRIMARY KEY or a UNIQUE index, the old row is deleted before the new row is inserted. |

# 4. COUNT()

|  |
| --- |
| SELECT COUNT(\*) FROM Products;  SELECT COUNT(COLUMN) FROM TABLE; |

# 5. SELECT DISTINCT

|  |
| --- |
| select distinct termsTarget from offer;  1 2 3 4 14 15 23 24 |

# 6.REPLACE: Find and replace string

|  |
| --- |
| MySQL UPDATE and REPLACE Without the example fieldnames and content to change, the query looks like this:  UPDATE [tablename]  SET [fieldname] = REPLACE([fieldname], 'text to find', 'text to replace with')  WHERE [fieldname] LIKE '%text to find%'  ‘text to find’ and ‘text to replace with’ need to be specific can not use: % |