MySQL Documentation

1. Logical operators

AND operator

```
SELECT *
FROM table_name
WHERE condition1 AND condition2 AND condition3...;
```

OR operator

```
SELECT *
FROM table_name
WHERE condition1 OR condition2 OR condition3...;
```

NOT operator

```
SELECT *
FROM table_name
WHERE NOT condition;
```

IN operator

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

BETWEEN operator

```
SELECT *
FROM table_name
WHERE column_name
BETWEEN value1 AND value2;
```

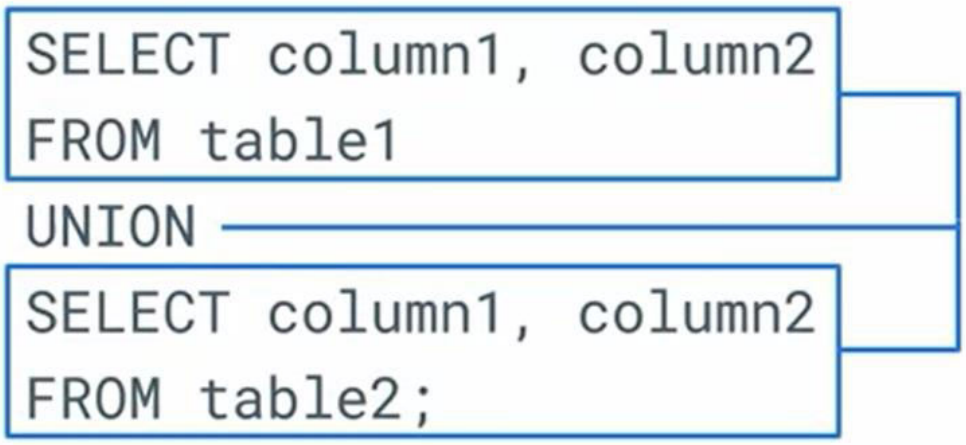
Start value End value

LIKE operator

```
SELECT *
FROM table_name
WHERE column_name
LIKE pattern;
```

_ (underscore) represents one single character whereas %(percent) represents 0,1 or more characters

UNION operator syntax



2. Alias

SQL alias syntax: renaming tables and columns

```
SELECT column1_name AS column1_alias, column2, column 3,
column4_name AS column4_alias
FROM table name;
```

SQL alias syntax: functions

```
SELECT CONCAT (column1, " ", column2) AS 'new_column_name'
FROM table_name
```

SQL alias syntax: multiple tables

```
SELECT x.column1, x.column2, y.column1,y.column2
FROM table_1 AS x, table_2 AS y
WHERE x.column2 < 12 AND y.column2 < 5;
```

3. Joins: Inner, Left, Right, Self

Inner JOIN syntax

```
SELECT table1_name.column_name
FROM table1_name
INNER JOIN table2_name
ON table1.column_name =  table2.column_name;
```

Left JOIN syntax

```
SELECT
table1alias.column1name AS "column 1 new name",
table1alias.column2name AS "column 2 new name",
table2alias.column1name AS "column 1 new name",
table2alias.column2name
FROM table1name AS table1alias
LEFT JOIN table2name AS table2alias
ON table1alias.column1name = table2alias.column1name;
```

RIGHT JOIN syntax

```
SELECT
table1alias.column1name AS "column 1 new name",
table1alias.column2name AS "column 2 new name",
table2alias.column1name AS "column 1 new name",
table2alias.column2name
FROM table1name AS table1alias
RIGHT JOIN table2name AS table2alias
ON table1alias.column1name = table2alias.column1name;
```

4. Grouping data: Group By

GROUP BY syntax

```
SELECT column1_name, column2_name
FROM table_name
WHERE filter_condition
GROUP BY column1_name, column2_name;
```

GROUP BY syntax with aggregate function

```
SELECT column1, column2, column3,     MAX(column1)
FROM table_name
GROUP BY column1, column2, column3;
```

5. Having

SQL SELECT statement syntax

```
SELECT column_name(s)
FROM table_name
WHERE filter_condition
GROUP BY group_by_column_or_expression
HAVING group_filter_condition;
```

6. Any / All

The syntax of a statement that uses an ANY operator is as follows:

```
SELECT column_name(s)
FROM table_name
WHERE column_name comparison operator ANY
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

The syntax for the ALL operator is as follows:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
  (SELECT column_name FROM table_name WHERE condition);
```

---

1. replace

REPLACE command

```
REPLACE INTO table_name (column1, column2, column3)
VALUES (column1_value, column2_value, column3_value);
```

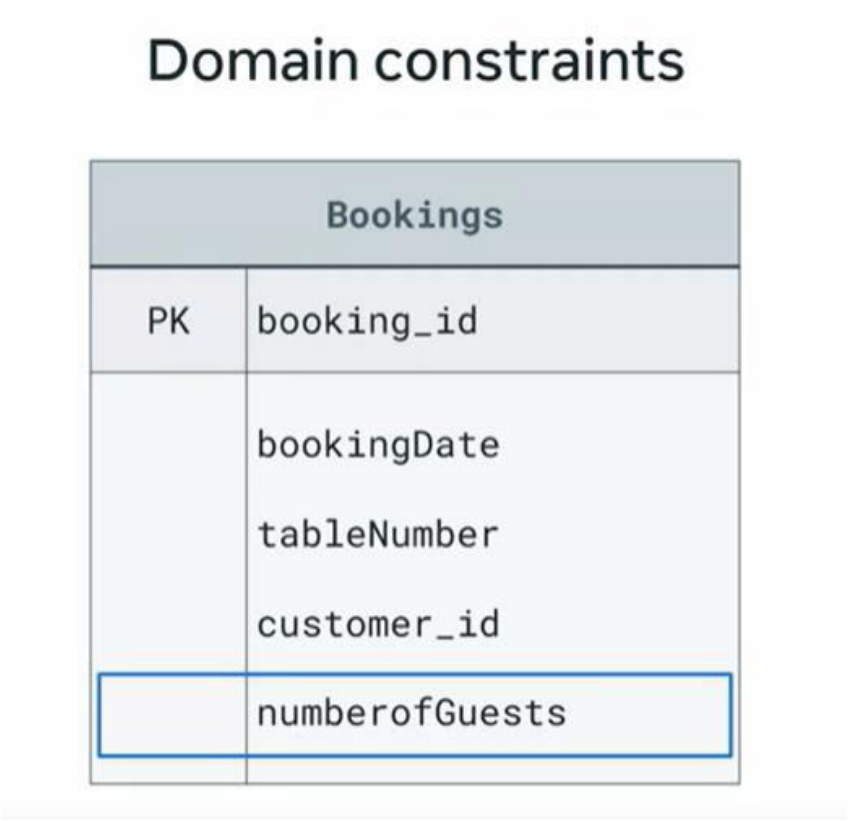
REPLACE command

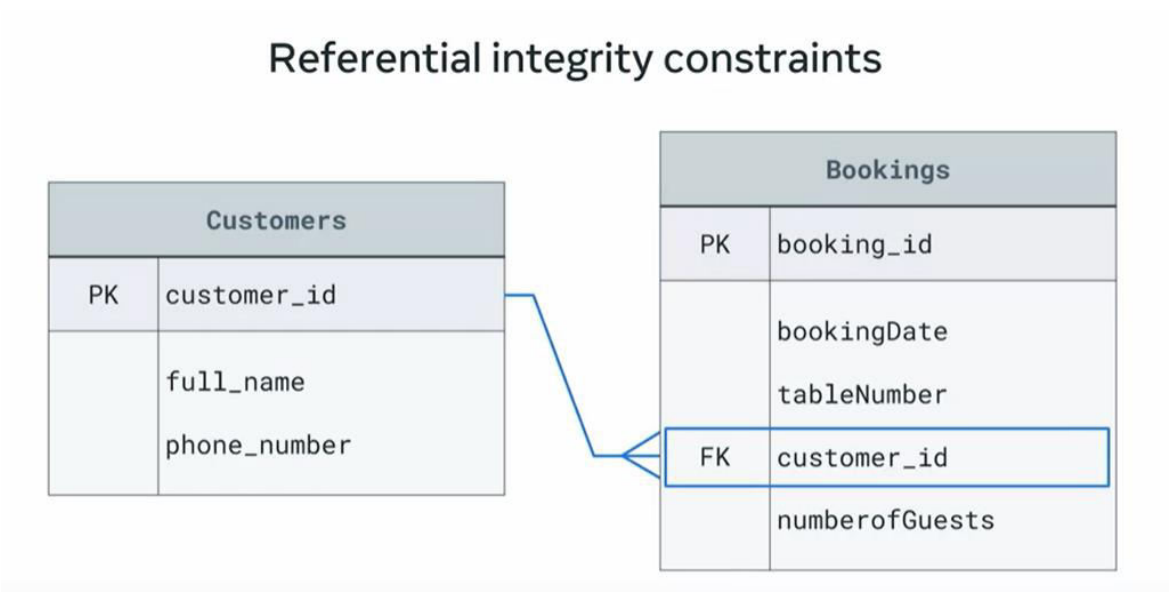
```
REPLACE INTO table_name (column1, column2, column3)
SET column_name = new_value;
```

2. constraints

Key constraints

Table_name	
PK	column1_name
	column2_name
	column3_name


Domain constraints
Bookings
PK
booking_id
bookingDate
tableNumber
customer_id
numberofGuests


Referential integrity constraints
Customers
PK
customer_id
full_name
phone_number
Bookings
PK
booking_id
bookingDate
tableNumber
FK
customer_id
numberofGuests

3. alter

ALTER TABLE statement

```
ALTER TABLE table_name
MODIFY column1_name VARCHAR(10) NOT NULL
MODIFY column2_name CHAR(10) NOT NULL
MODIFY column3_name INT NOT NULL;
```

ALTER TABLE statement

```
ALTER TABLE table_name
ADD COLUMN column_name;
```

ALTER TABLE statement

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

4. changing table structure

CREATE TABLE syntax

```
CREATE TABLE new_table_name
SELECT columns
FROM existing_table_name;
```

different databases:

CREATE TABLE syntax

```
CREATE TABLE database_X_name.new_table_name
SELECT columns
FROM database_Y_name.existing_table_name;
```

to be copied with constraints

```
create table ClientsTest3  like Clients;
```

CHANGE

You can use the CHANGE command with the ALTER statement to rename a column.

Syntax:

```
ALTER TABLE table_name CHANGE from_column to_column datatype;
```

Example:

```
ALTER TABLE Employees CHANGE Email BusinessEmail VARCHAR(50);
```

RENAME

The RENAME command can be used to change a table name, followed by the new name that needs to be given to the table.

Syntax:

```
ALTER TABLE table_name RENAME new_table_name;
```

Example:

```
ALTER TABLE OrderStatus RENAME OrderDeliveryStatus;
```

5. Subquery

Subquery syntax

OUTER / PARENT QUERY

```
SELECT column_name(s)
FROM table_name
WHERE expression operator
(SELECT column_name FROM table_name WHERE condition);
```

INNER / CHILD QUERY

EXISTS operator syntax

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

6. VIEWS

VIEW syntax

```
CREATE VIEW view_name AS
SELECT table1.column1, table1.column2
FROM table_name
WHERE condition;
```

Dot notation syntax

```
CREATE VIEW view_name AS alias
SELECT table1.column1, table1.column2, table2.column1, table2.column2
FROM table1_name INNER JOIN table2_name
ON table1.column1 = table2.column1
WHERE condition;
```

Renaming Virtual Tables

In case you want to rename the virtual table, you can do that exactly in the same way you rename a normal table.

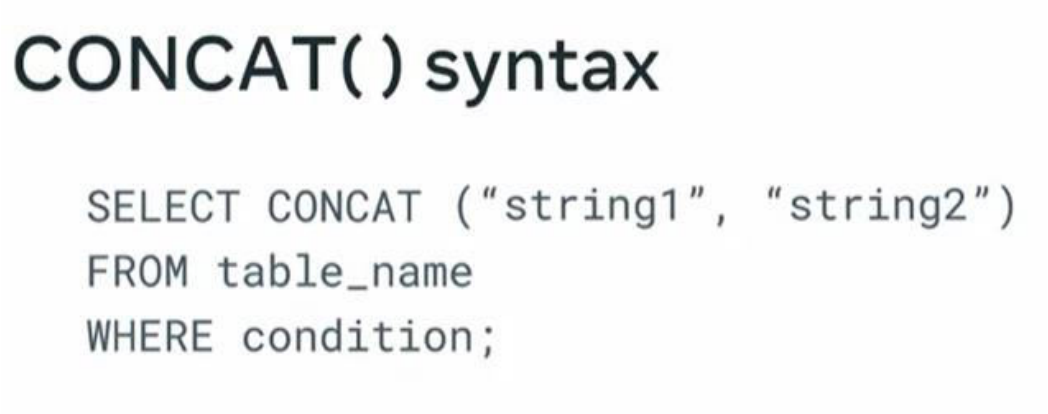
Syntax:

```
1   RENAME TABLE VirtualTableName TO NewVirtualTableName;
```

FUNCTIONS

1. Math functions (numeric) - round(), mod()

2. String functions - concat(), substr(), lcase(), ucase()



SUBSTR( ) syntax

```
SELECT SUBSTR  (“string”, start index, length)
FROM table_name
WHERE condition;
```

UCASE( ) syntax

```
SELECT UCASE(column_name)
FROM table_name;
```

LCASE( ) syntax

```
SELECT LCASE(column_name)
FROM table_name;
```

3. Date functions:

CURRENT_DATE( ) syntax

```
SELECT CURRENT_DATE();
```

CURRENT_TIME( ) syntax

```
SELECT CURRENT_TIME();
```

DATE_FORMAT( ) syntax

```
DATE_FORMAT('YYY-MM-DD', "format")
```

DATEDIFF( ) syntax

```
SELECT DATEDIFF("date_1", "date_2");
```

4. Comparison operators:

GREATEST( ) and LEAST( ) syntax

```
SELECT column1
GREATEST (column2, column3, column4) AS highest,
LEAST (column2, column3, column4) AS lowest,
FROM table_name;
```

ISNULL( ) with SELECT clause

```
SELECT ISNULL(column_name)
FROM table_name
```

ISNULL( ) with WHERE clause syntax

```
SELECT *
FROM table_name
WHERE ISNULL(column_name)
```

5. control flow functions:

CASE syntax

```
SELECT column_name
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END AS alias
FROM table;
```

2. PROCEDURES

Stored procedure basic syntax

```
CREATE PROCEDURE ProcedureName()
SELECT column_name
FROM table_name;
```

Stored procedure with parameter

```
CREATE PROCEDURE ProcedureName(parameter_value INT)
SELECT column_name
FROM table_name
WHERE value = parameter_value;
```

to invoke it:

Call stored procedure

```
CALL ProcedureName();
```

Drop stored procedure

```
DROP PROCEDURE ProcedureName;
```