DDL Commands

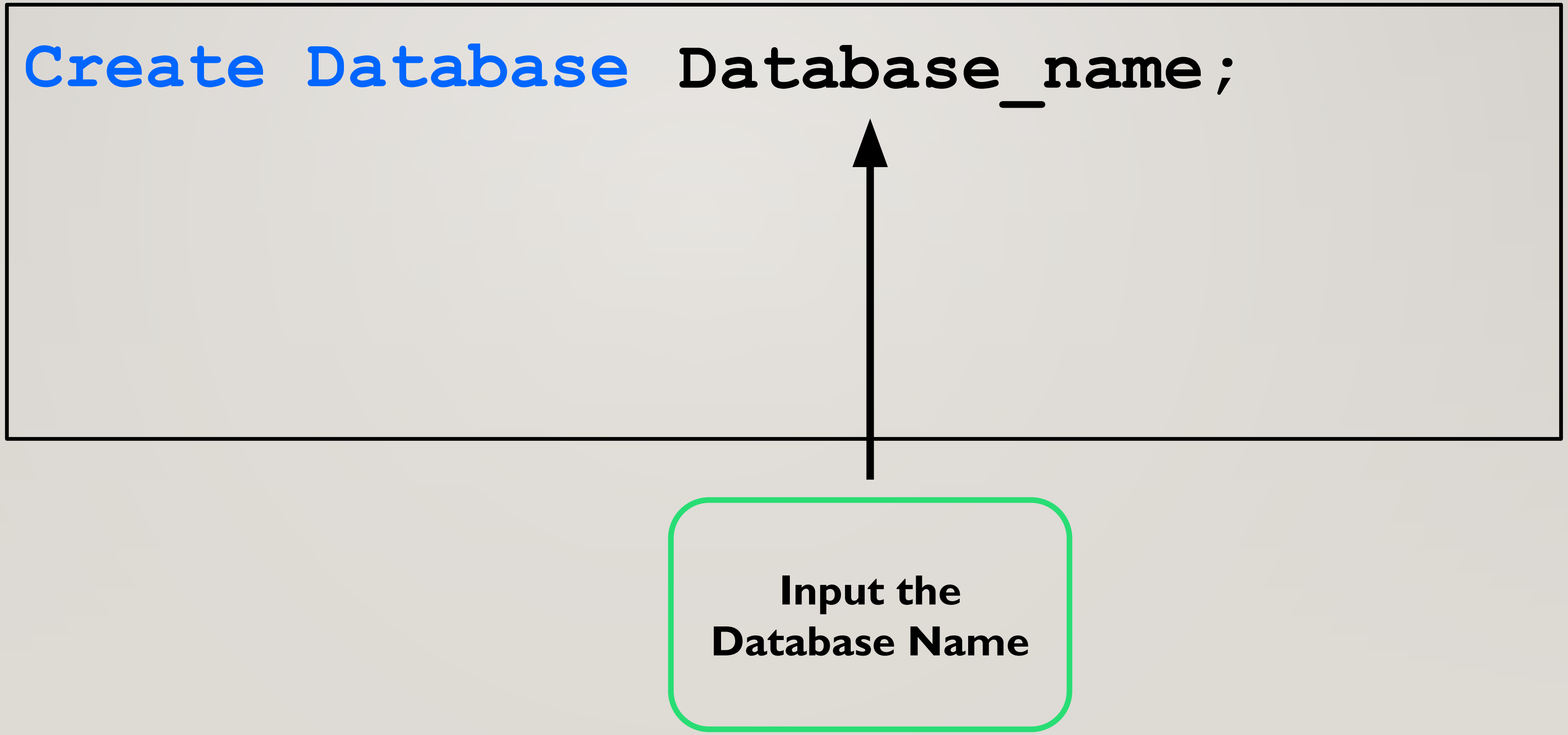
CREATE TABLE

SYNTAX : CREATE DATABASE

```
Create Database
```

Use Create Database command

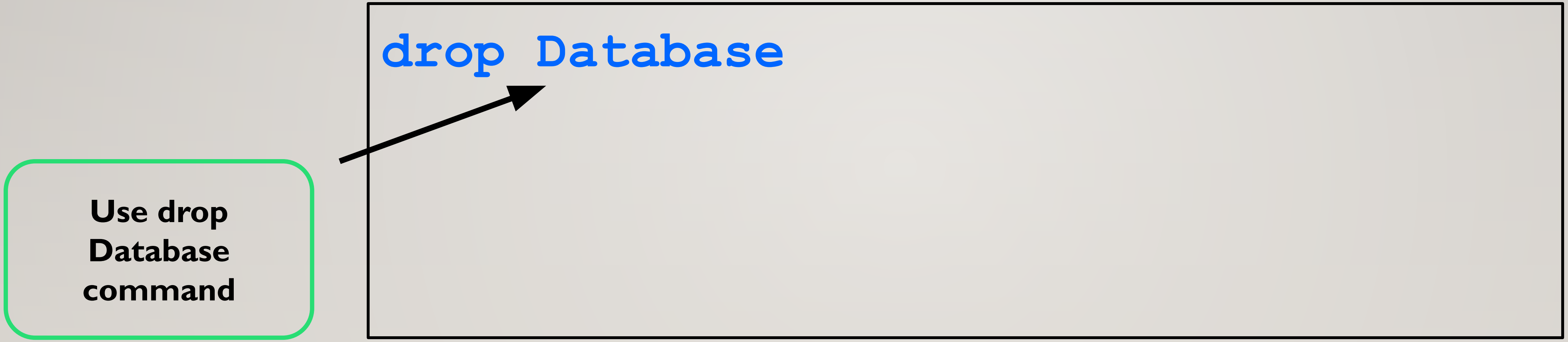
SYNTAX : CREATE DATABASE



```
Create Database Database_name;
```

Input the
Database Name

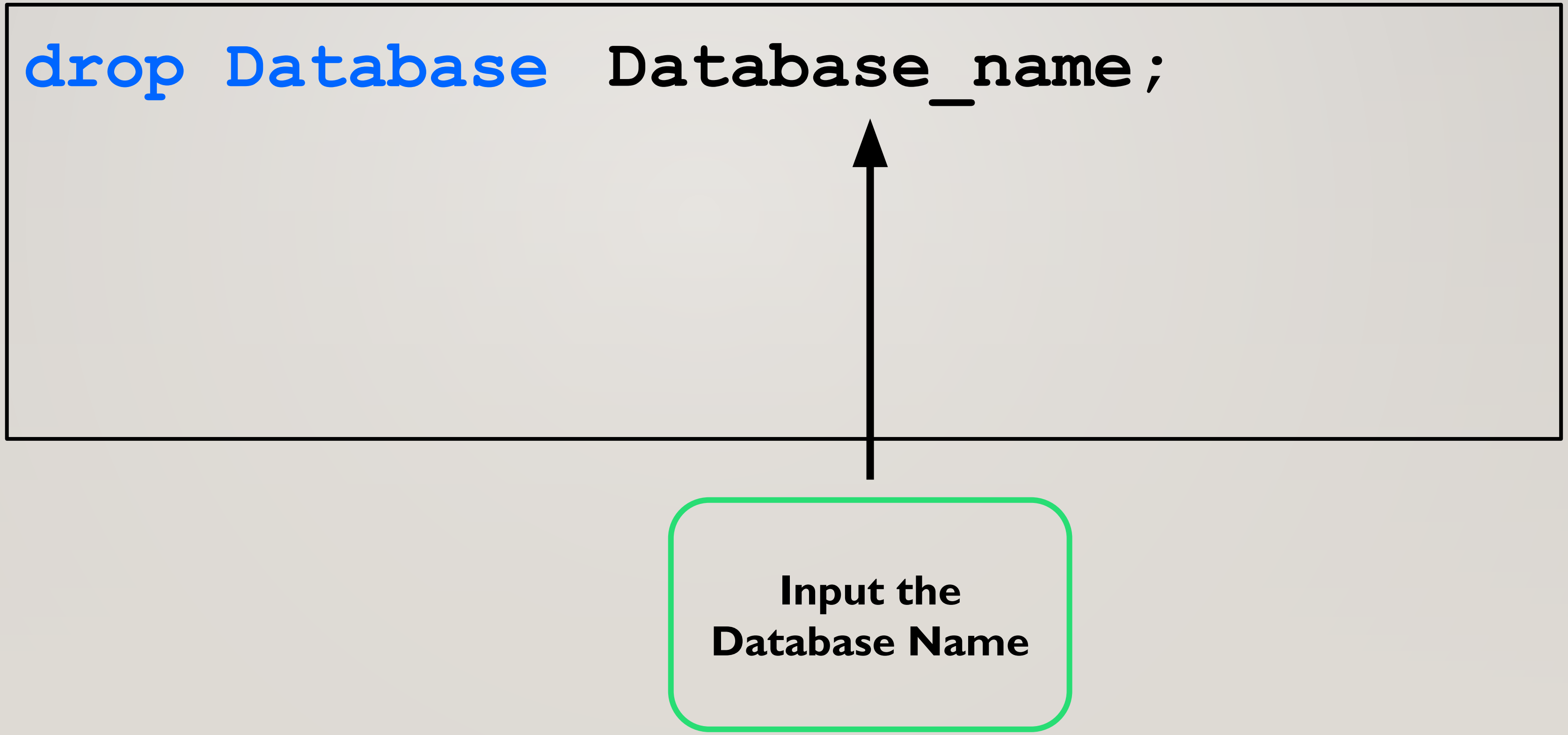
SYNTAX : CREATE DATABASE



```
drop Database
```

Use drop Database command

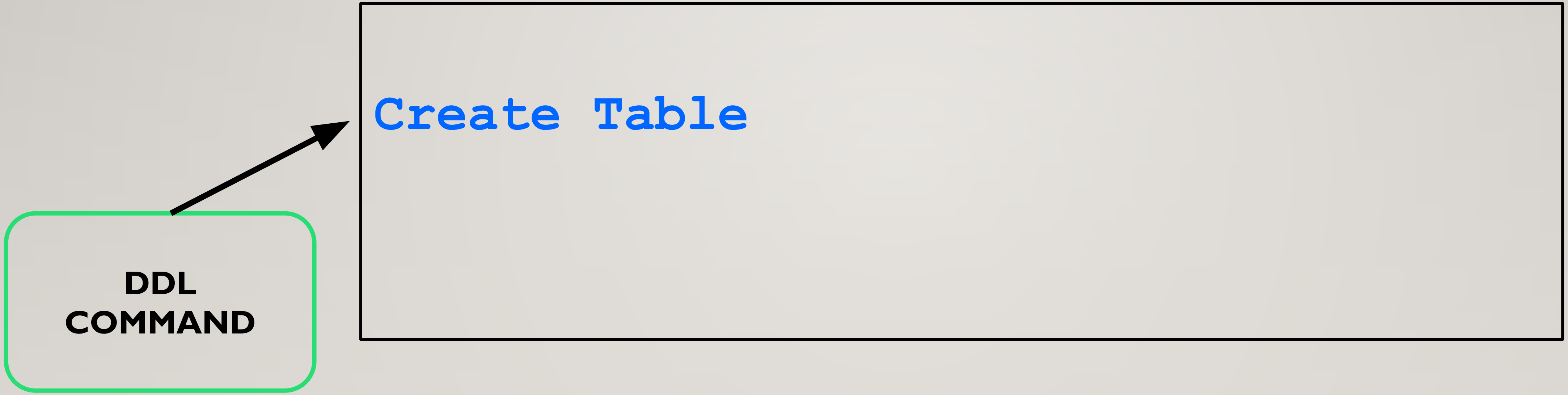
SYNTAX : CREATE DATABASE



```
drop Database  Database_name;
```

Input the Database Name

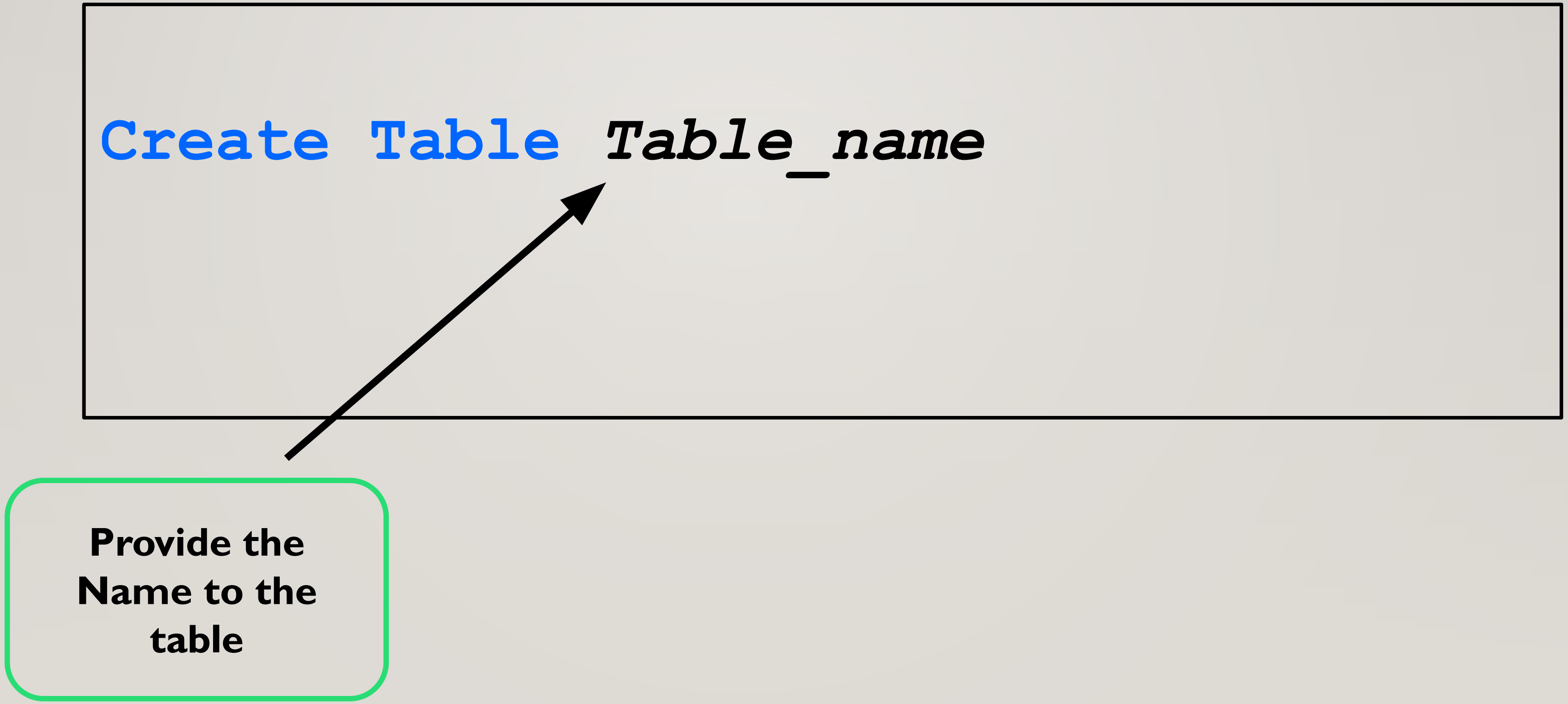
SYNTAX : CREATE TABLE



```
Create Table
```

DDL COMMAND

SYNTAX : CREATE TABLE



```
Create Table Table_name
```

Provide the Name to the table

SYNTAX : CREATE TABLE

```
Create Table Table_name
(Column_name
```

Provide the Column Details starting with the Column Name

SYNTAX : CREATE TABLE

```
Create Table Table_name
(Column_name data_type() Constraints
```

Define Constraints; If Any

SYNTAX : CREATE TABLE

```
Create Table Table_name
(Column_name data_type()
```

Define the Data Type and Number of Characters

SYNTAX : CREATE TABLE

```
Create Table Table_name
(Column_name data_type() Constraints
Column2_name data_type() Constraints
```

Repeat for the next column

SYNTAX : CREATE TABLE

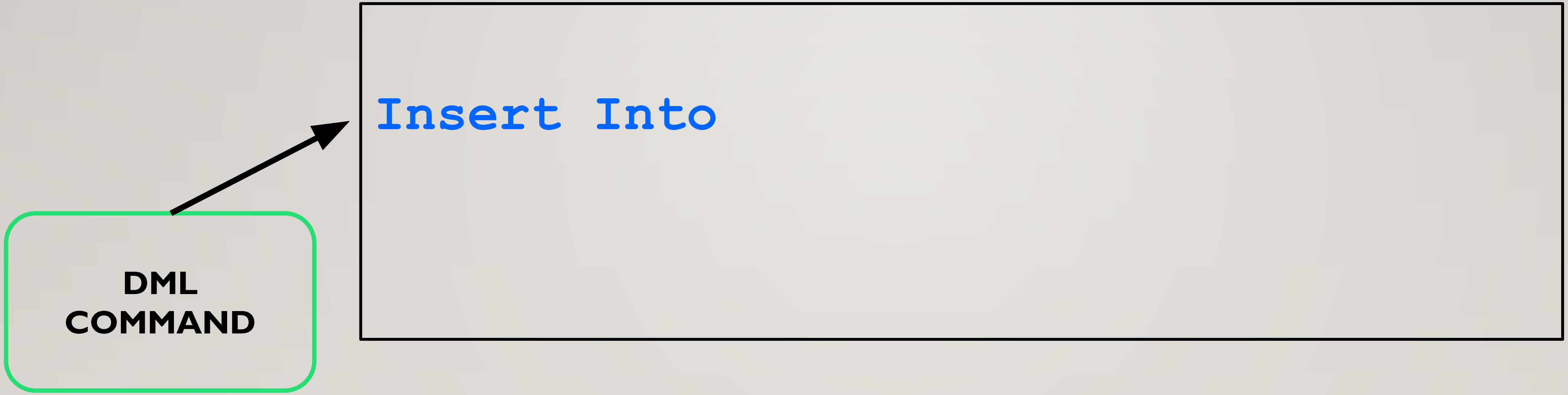
```
Create Table Table_name
(Column_name data_type() Constraints
Column2_name data_type() Constraints
             ...
             ...                      )
```

Close Parenthesis after Defining all Columns

DML Commands

INSERTING DATA

SYNTAX : INSERT INTO



```
Insert Into
```

DML COMMAND

SYNTAX : INSERT INTO

```
Insert Into table_name
```

Input the Table Name

SYNTAX : INSERT INTO

```
Insert Into table_name
Values (
column1_value,column2_value,……
…., column_value);
```

VALUES
COMMAND

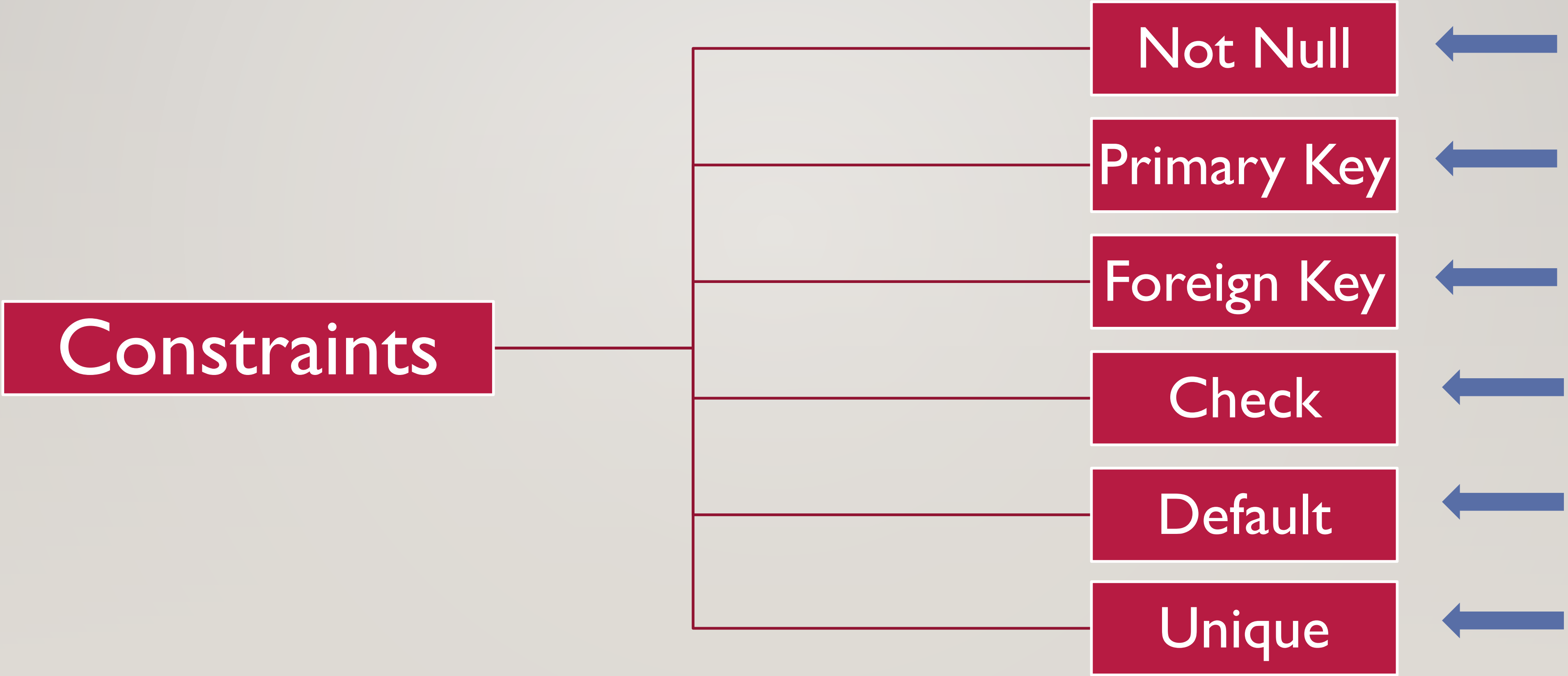
SYNTAX : INSERT INTO

```
Insert Into table_name
Values (
column1_value,column2_value,……
…., column_value);
```

Respective Values of all columns in order

DCL Commands

“CONSTRAINTS”



Constraints

Not Null

Primary Key

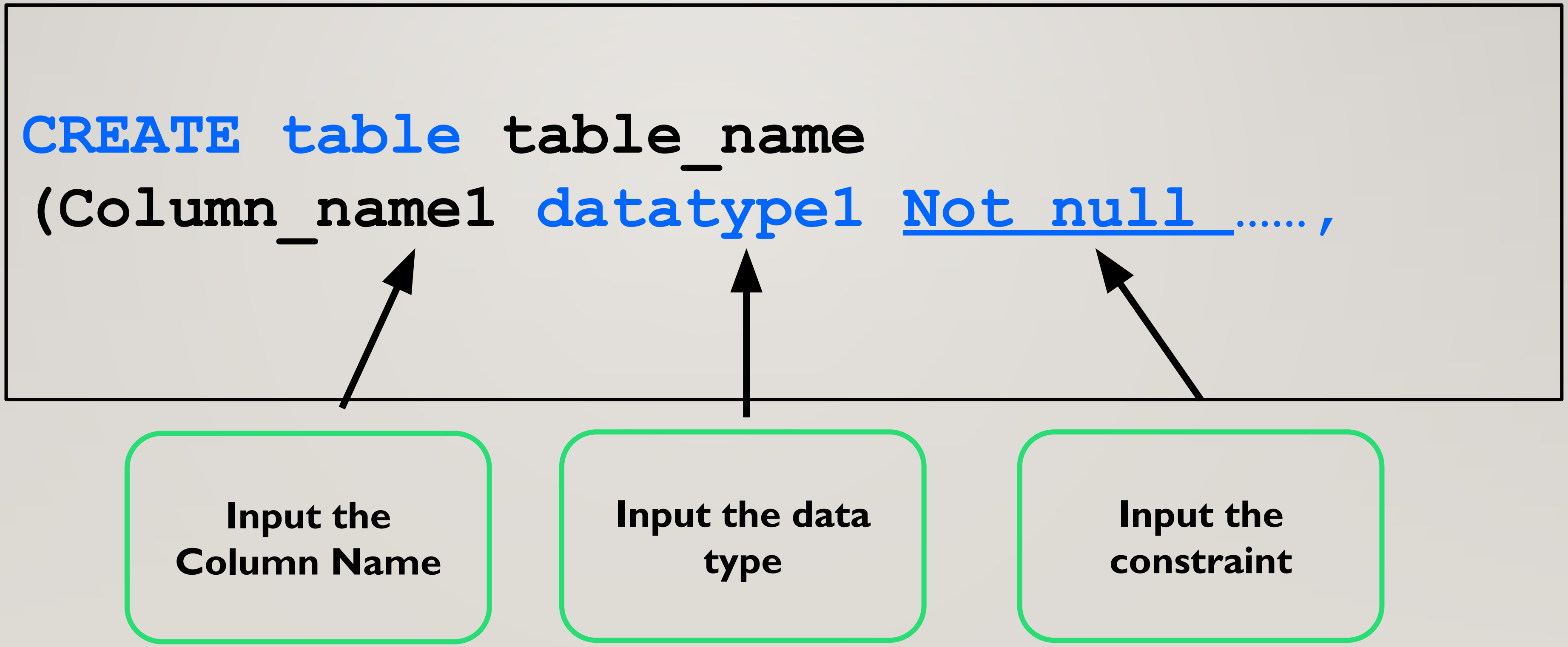
Foreign Key

Check

Default

Unique

SYNTAX : NOT NULL



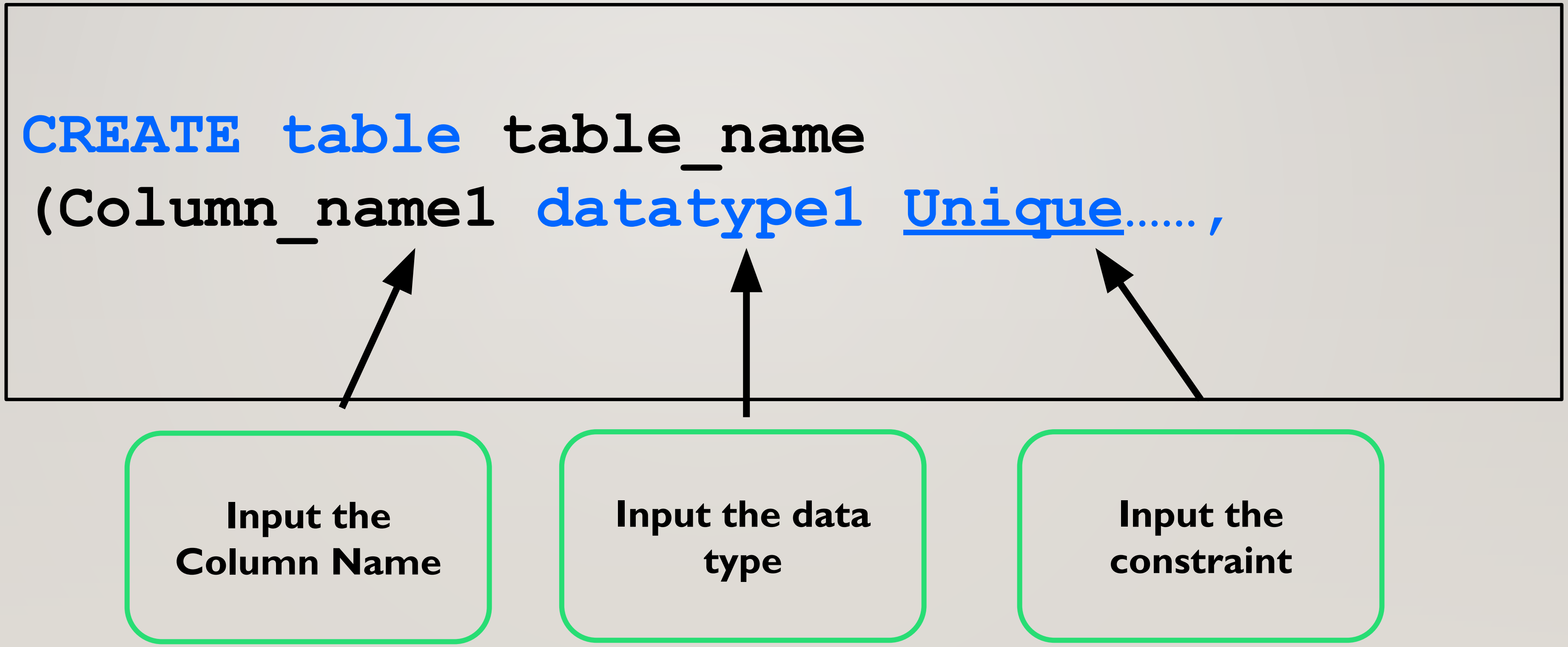
```
CREATE table table_name
(Column_name1 datatype1 Not null ......,
```

Input the Column Name

Input the data type

Input the constraint

SYNTAX : UNIQUE



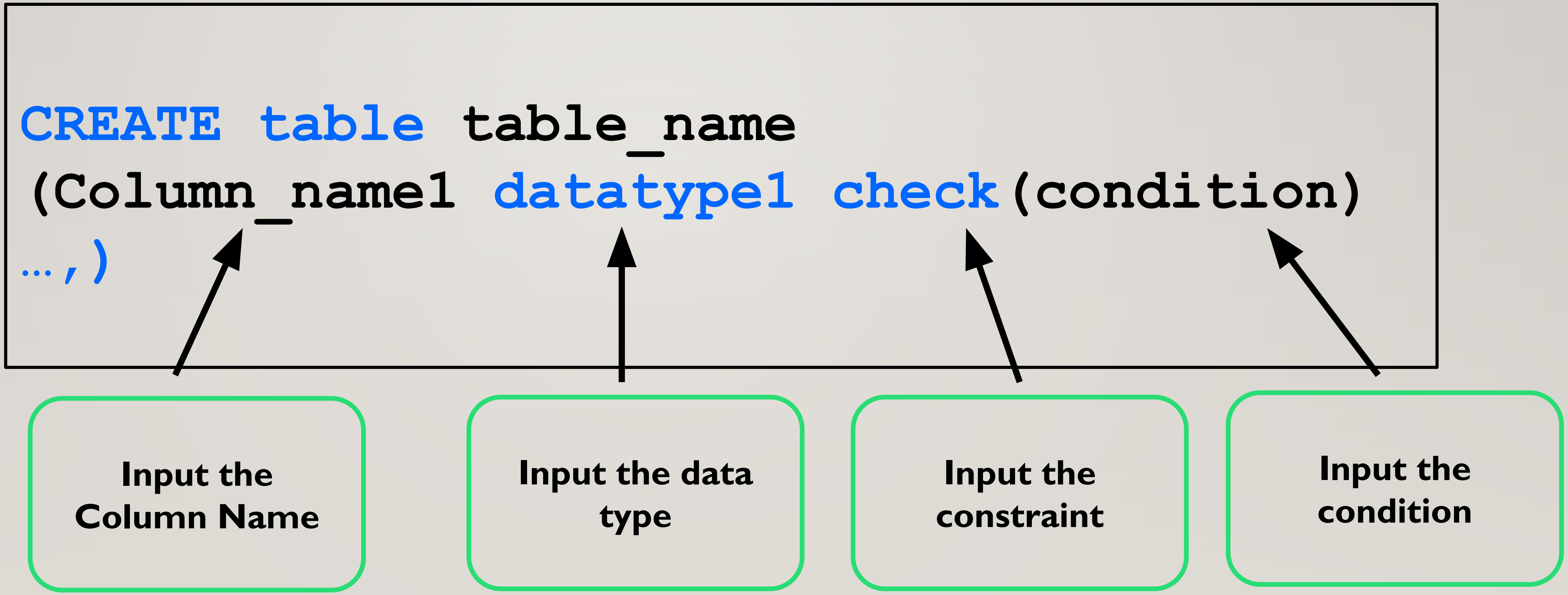
```
CREATE table table_name
(Column_name1 datatype1 Unique...... ,
```

Input the Column Name

Input the data type

Input the constraint

SYNTAX : CHECK



```
CREATE table table_name
(Column_name1 datatype1 check(condition)
…,)
```

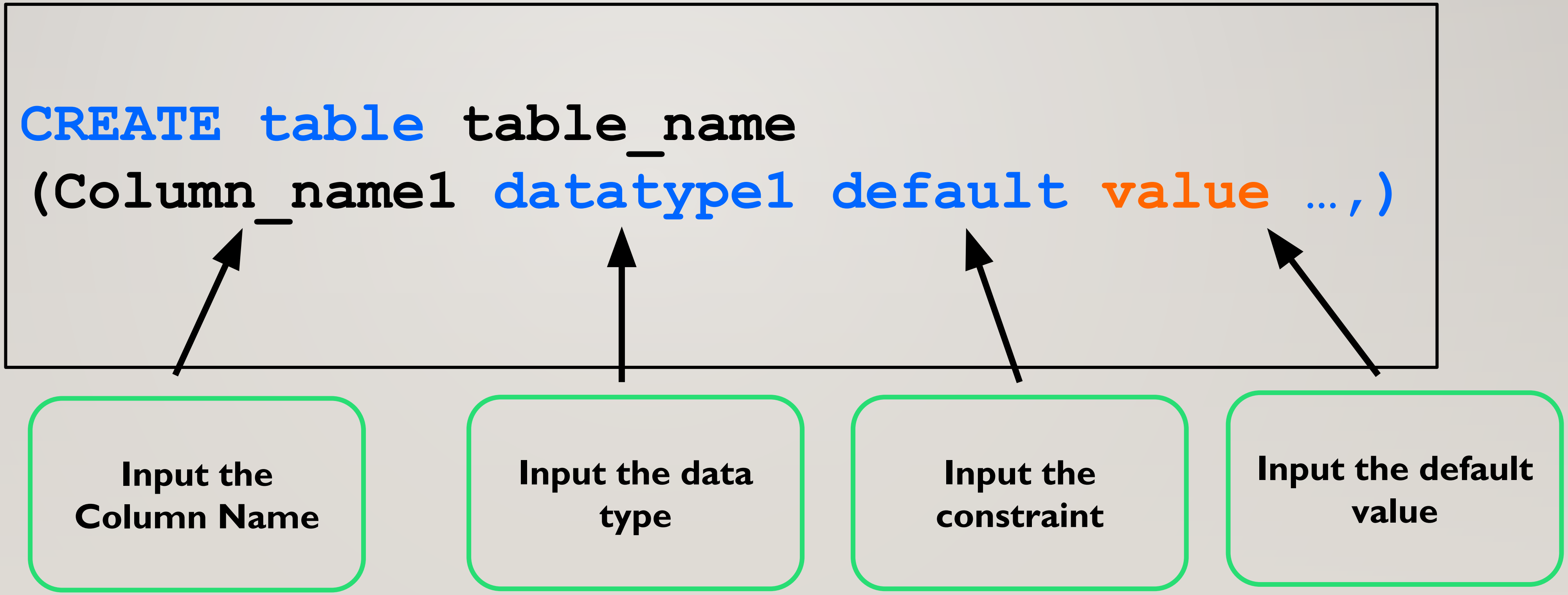
Input the Column Name

Input the data type

Input the constraint

Input the condition

SYNTAX : DEFAULT



```
CREATE table table_name
(Column_name1 datatype1 default value …,)
```

Input the Column Name

Input the data type

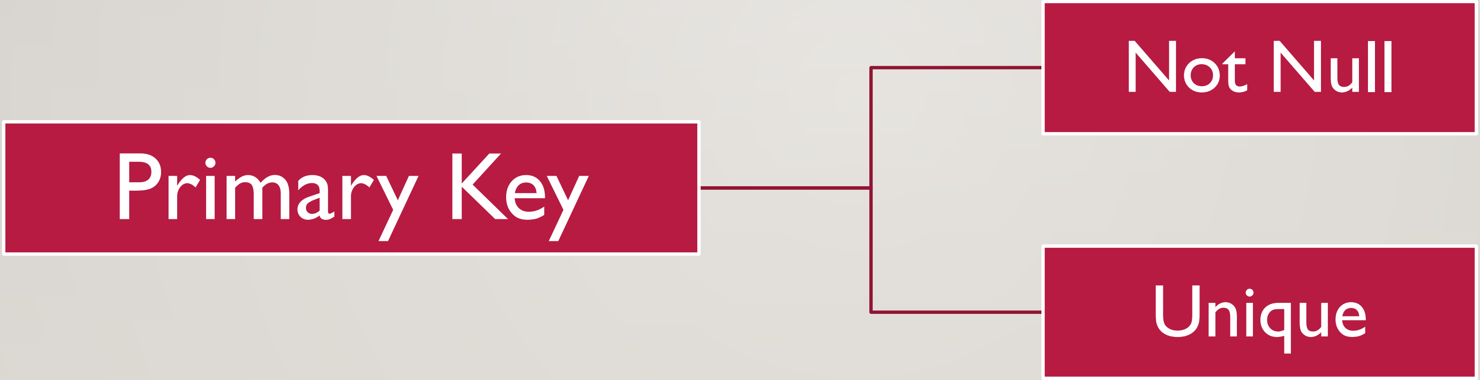
Input the constraint

Input the default value

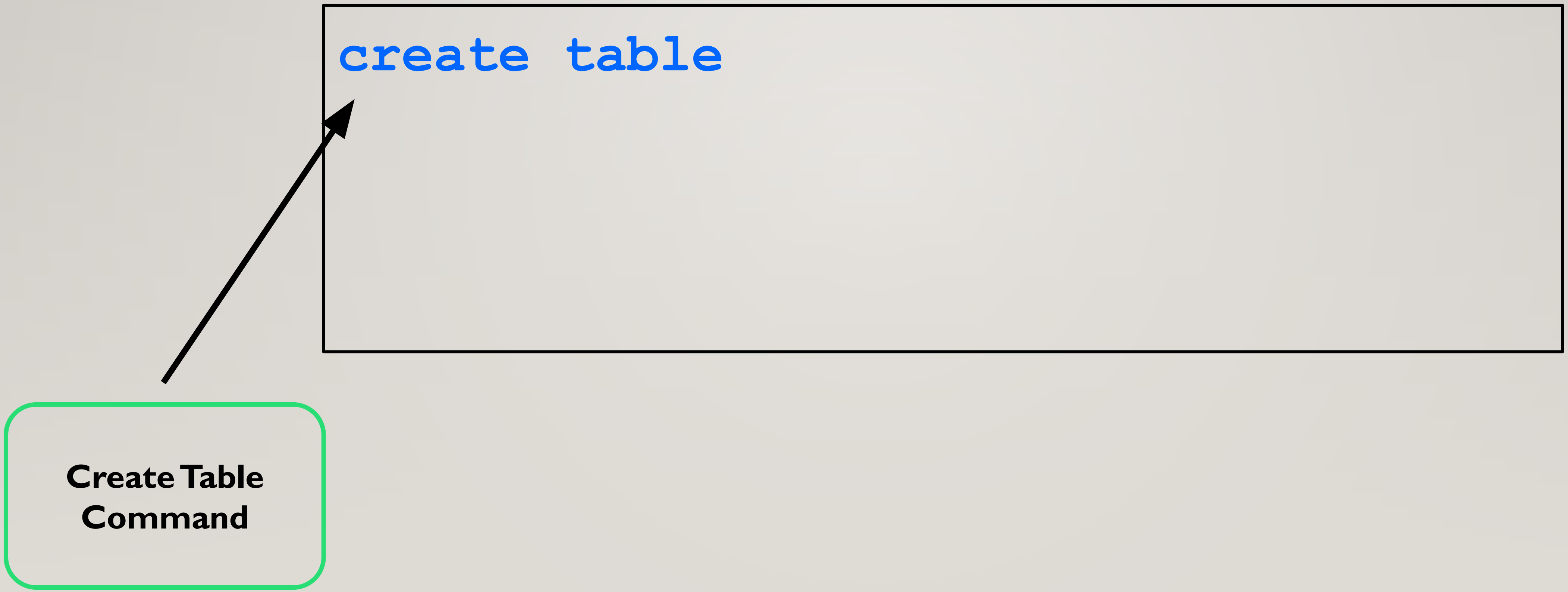
DDL Commands

PRIMARY KEY

SYNTAX : PRIMARY KEY



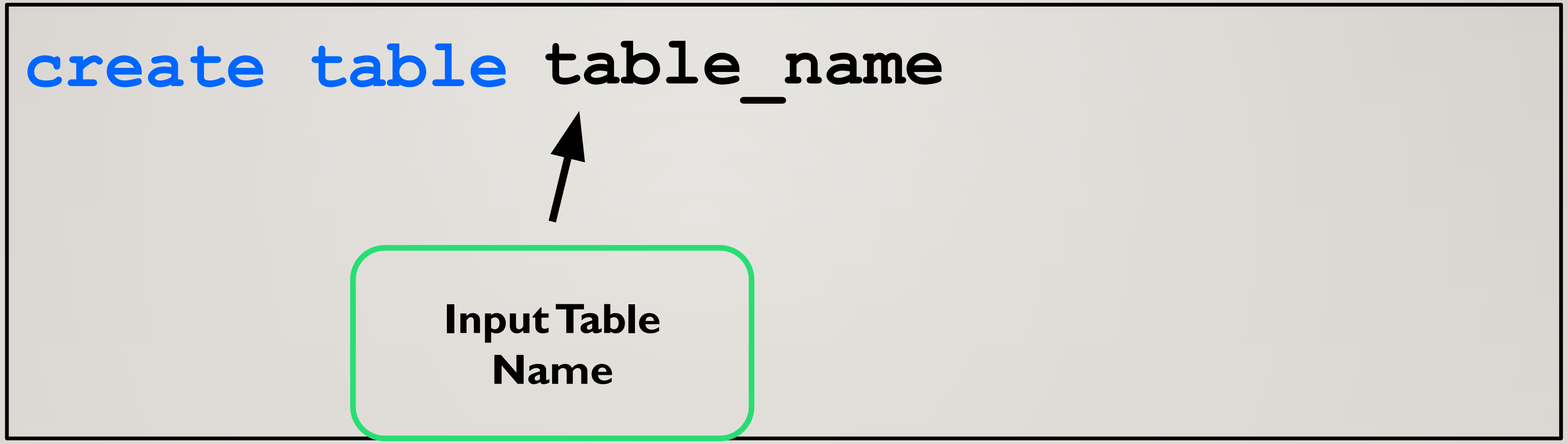
SYNTAX : PRIMARY KEY



```
create table
```

Create Table Command

SYNTAX : PRIMARY KEY



```
create table table_name
```

Input Table Name

SYNTAX : PRIMARY KEY

```
create table table_name
(Column_name1 datatype1,Column_name2
datatype2, ……,
```

Input the column names

SYNTAX : PRIMARY KEY

```
create table table_name(Column_name1
datatype1,Column_name2 datatype2, ......,
Primary key(Column_name1)
```

Use primary Key with the column name

SYNTAX : PRIMARY KEY

```
create table table_name(Column_name1
datatype1,Column_name2 datatype2, ……,
Primary key(Column_name1)
);
```

End with parenthesis

Table Relationships

Primary Key

Parent Table

Foreign Key

Child Table

Relationships Rules

The two tables should have a common column

Common Column data in Child table should be present in the parent table

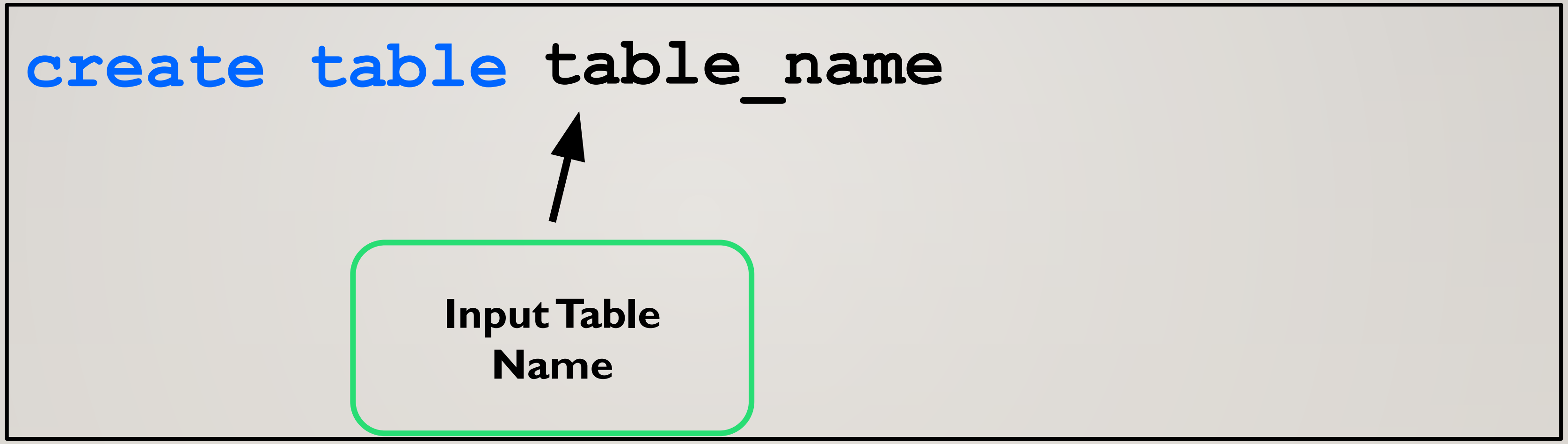
Data from Child table cannot be deleted before the parent data

SYNTAX : FOREIGN KEY

```
create table
```

Create Table Command

SYNTAX : FOREIGN KEY



```
create table table_name
```

Input Table Name

SYNTAX : FOREIGN KEY

```
create table table_name
(Column_name1 datatype1,Column_name2
datatype2, ……,
```

Input the column names

SYNTAX : FOREIGN KEY

```
create table table_name(Column_name1
datatype1,Column_name2 datatype2, ……,
foreign key(Column_name1) references
parent_table(Column_name)
```

Use forgein Key with the column name

SYNTAX : FOREIGN KEY

```
create table table_name(Column_name1
datatype1,Column_name2 datatype2, ……,
foreign key(Column_name1) references
parent_table(Column_name)
);
```

End with parenthesis

DATA TYPES

INT	**EXAMPLES** : 28, -6	Used for Non Decimal Numeric Data Ex – Age, Weight, Income,
CHAR	**EXAMPLES** : John, Marketing etc	Used for Fixed length Text Data Ex – Product Codes, Postal Codes etc
VAR CHAR	**EXAMPLES** : John,	Used for Variable Text Data Ex – First Name, last name etc.
FLOAT	**EXAMPLES** : 2.4 , -66.5	Used for Decimal Numeric Data Ex – Commission Percentage, etc

DATA TYPES

BOOLEAN	**EXAMPLES** :True/False, Male/Female	Used for Data with only 2 possible entries,
DATE	**EXAMPLES** : 24/02/2023	Used for Entering Date
TIME	**EXAMPLES** : 12:22	Used for Time
DATETIME	**EXAMPLES** : 23/10/2023 11:23	Used for Date and Time entries :Transaction Data

DQL Commands

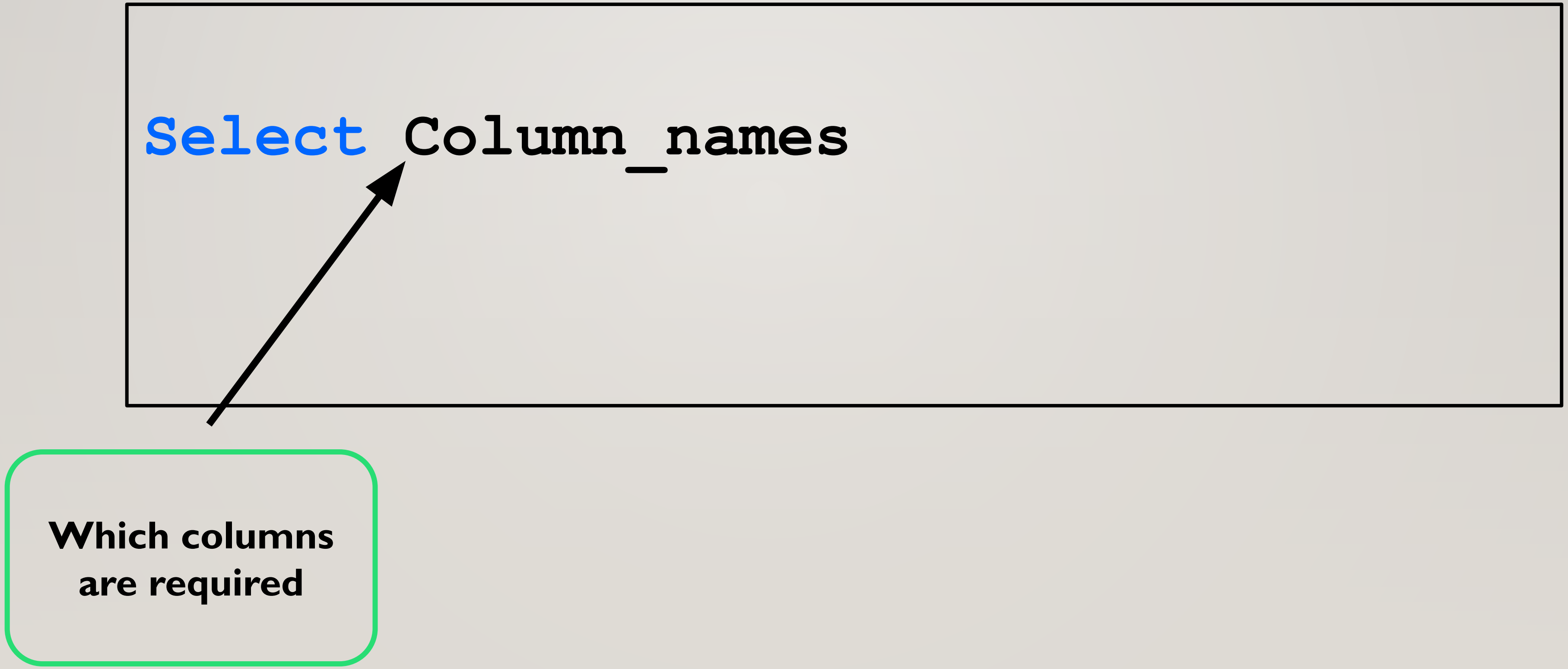
“SELECT”

SYNTAX : SELECT

```
Select
```

DQL COMMAND

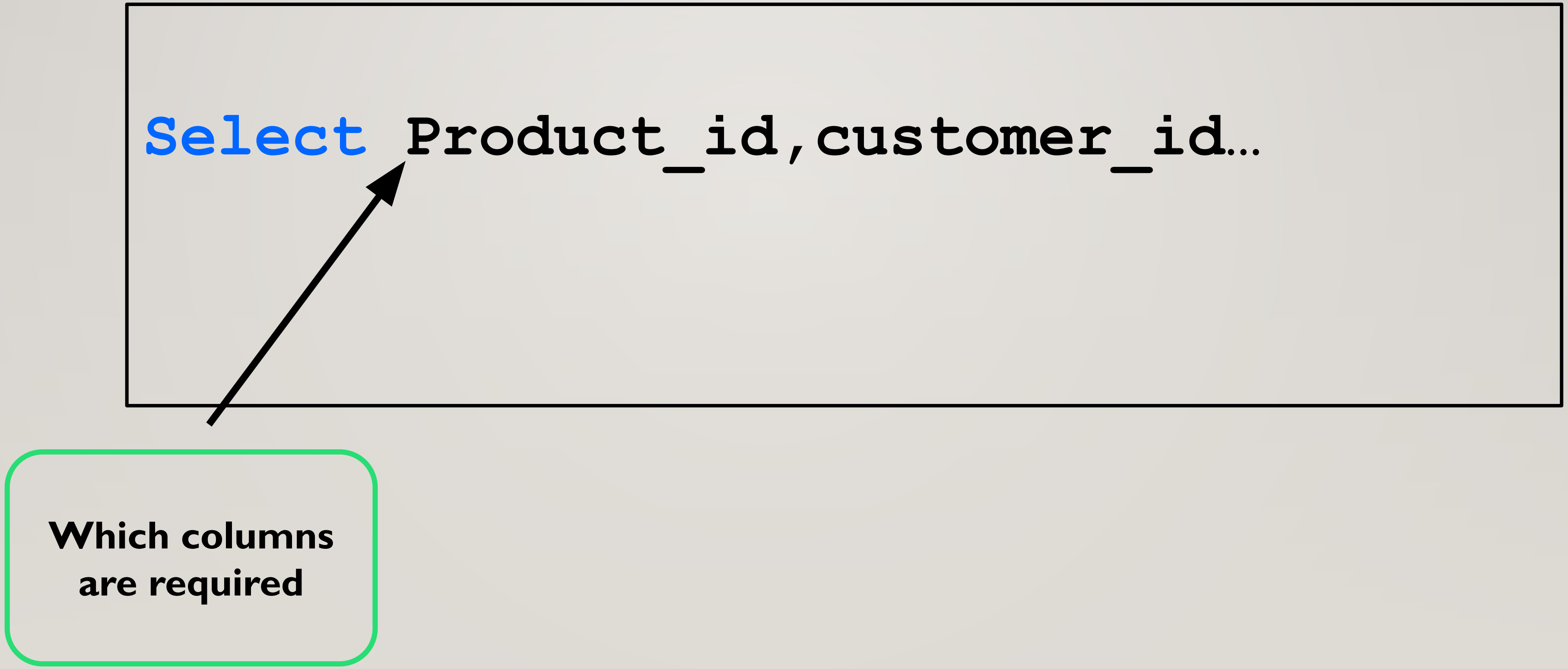
SYNTAX : SELECT



```
Select Column_names
```

Which columns are required

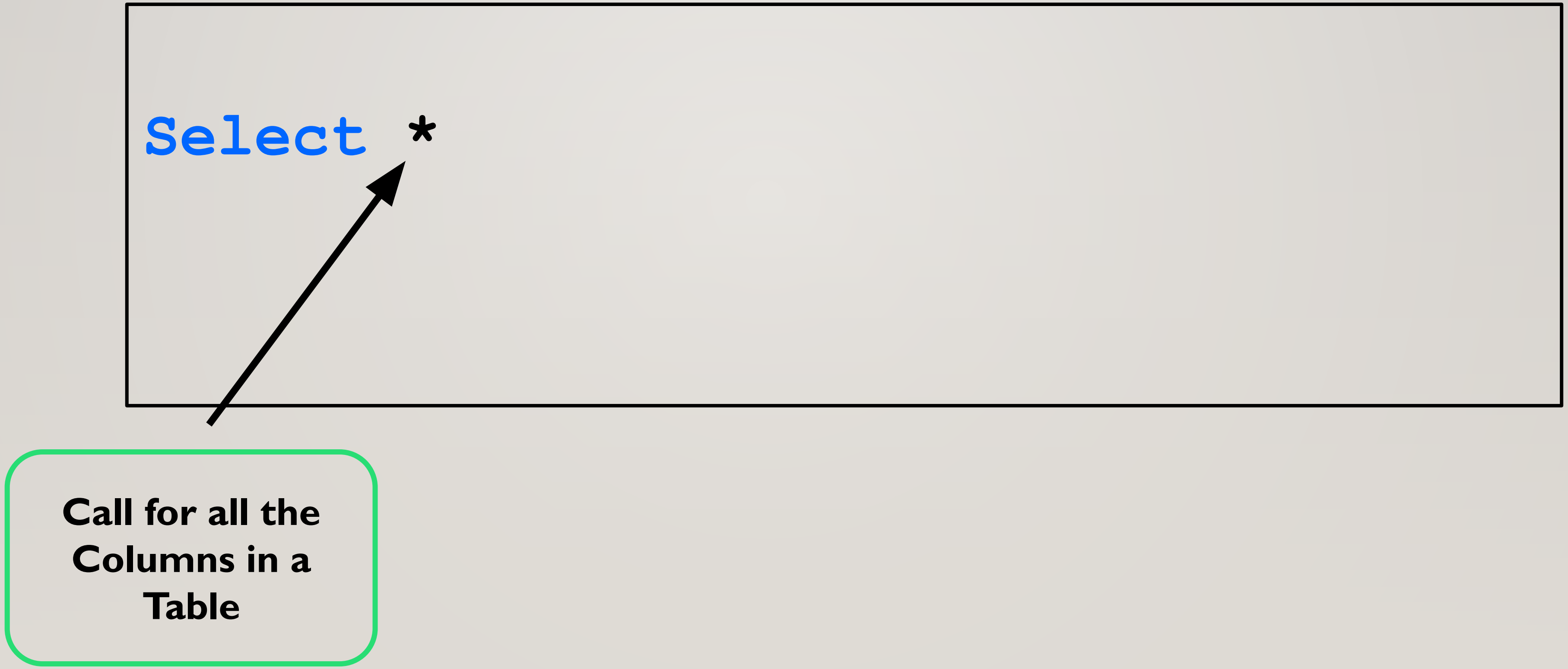
SYNTAX : SELECT



```
Select Product_id,customer_id…
```

Which columns are required

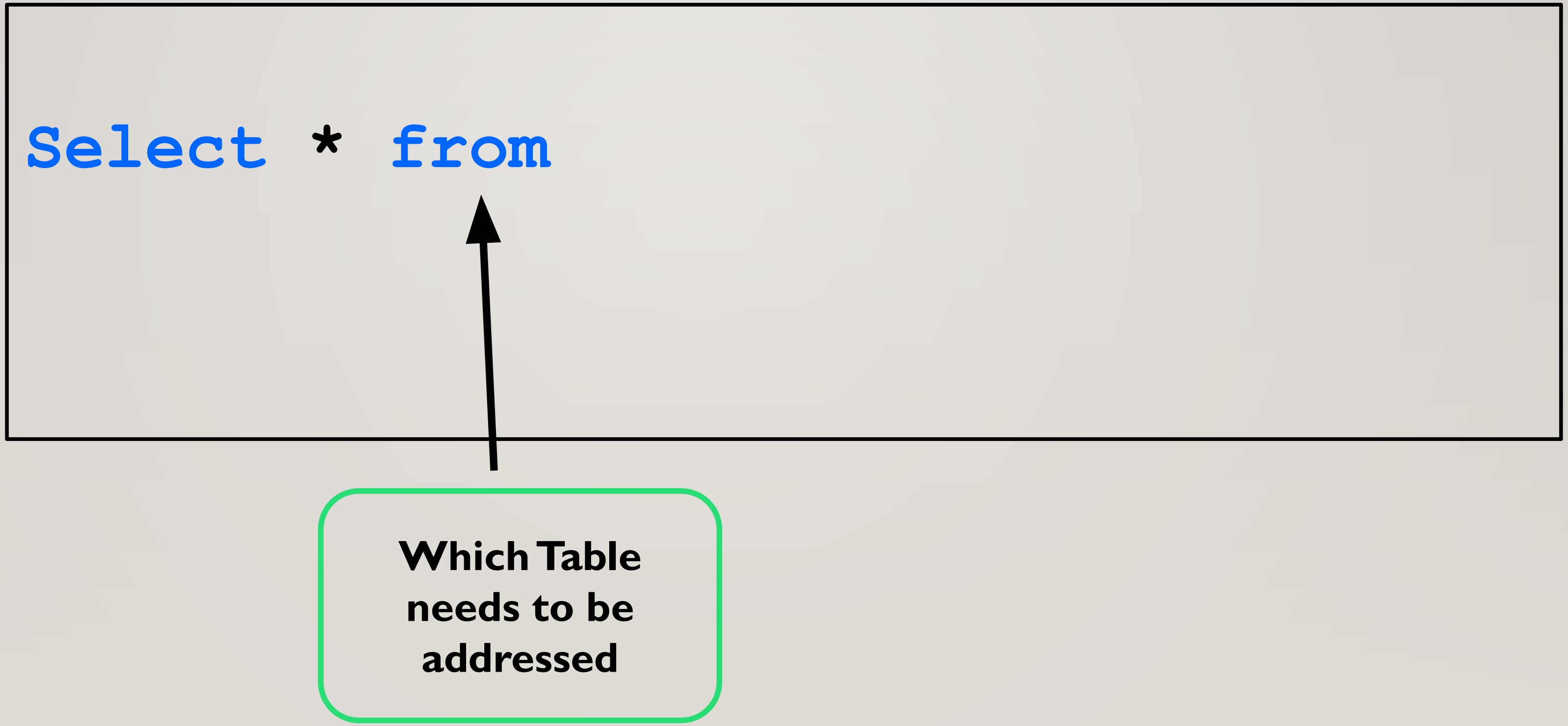
SYNTAX : SELECT



```
Select *
```

Call for all the Columns in a Table

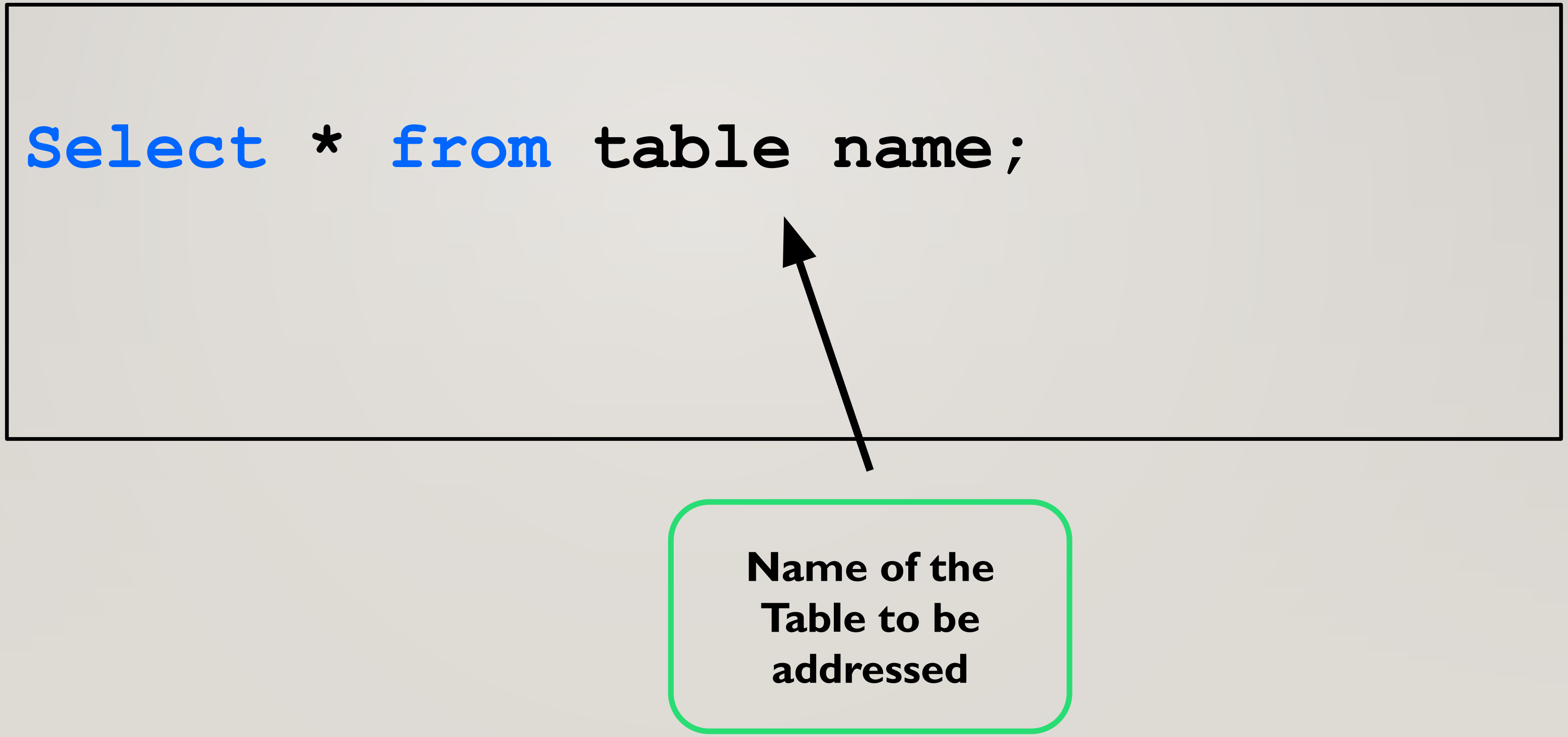
SYNTAX : SELECT



```
Select * from
```

Which Table needs to be addressed

SYNTAX : SELECT



```
Select * from table name;
```

Name of the Table to be addressed

SYNTAX : SELECT

```
Select Product_id,customer_id from products
```

SYNTAX : SELECT

```
Select * from products
```

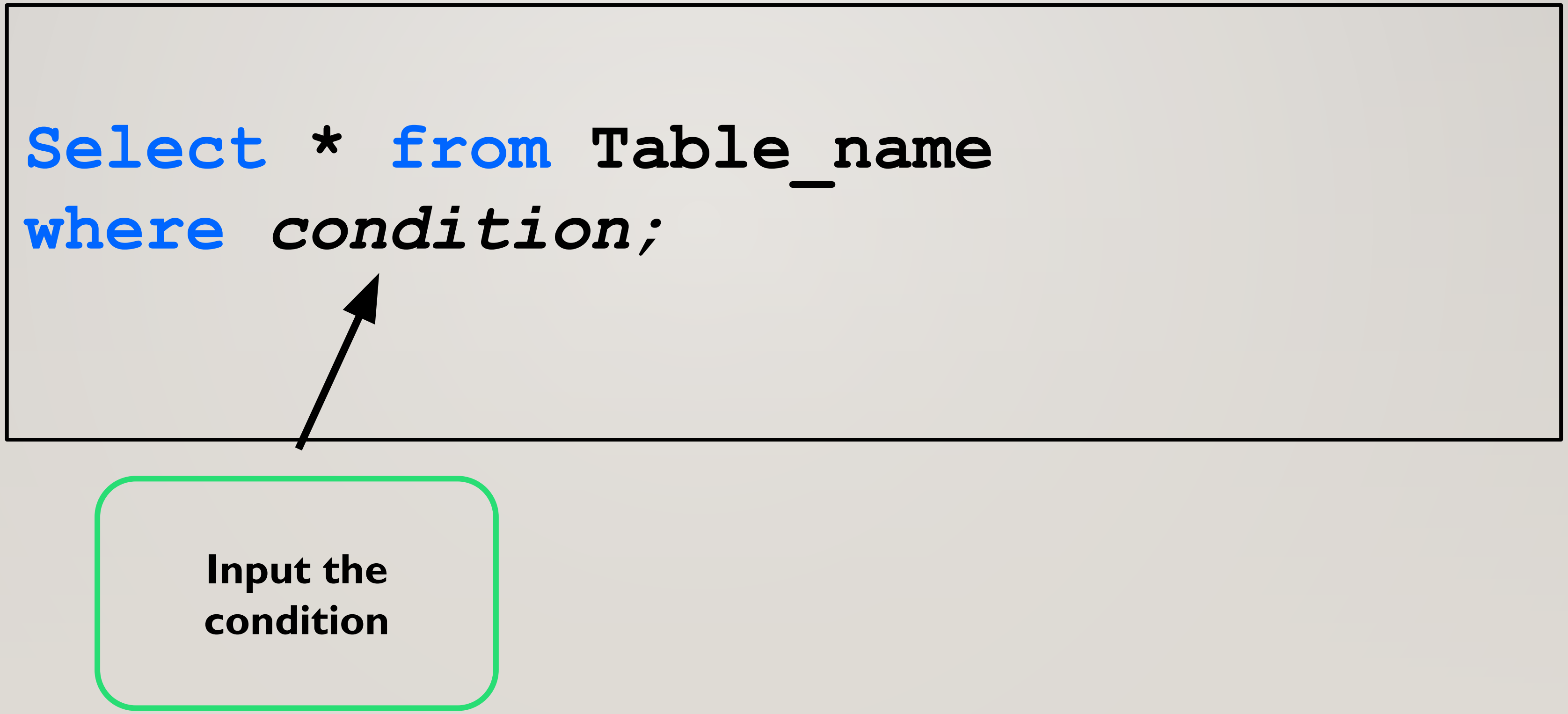
FILTERING THE DATA

SYNTAX : SELECT

```
Select * from Table_name
where
```

Where command

SYNTAX : SELECT



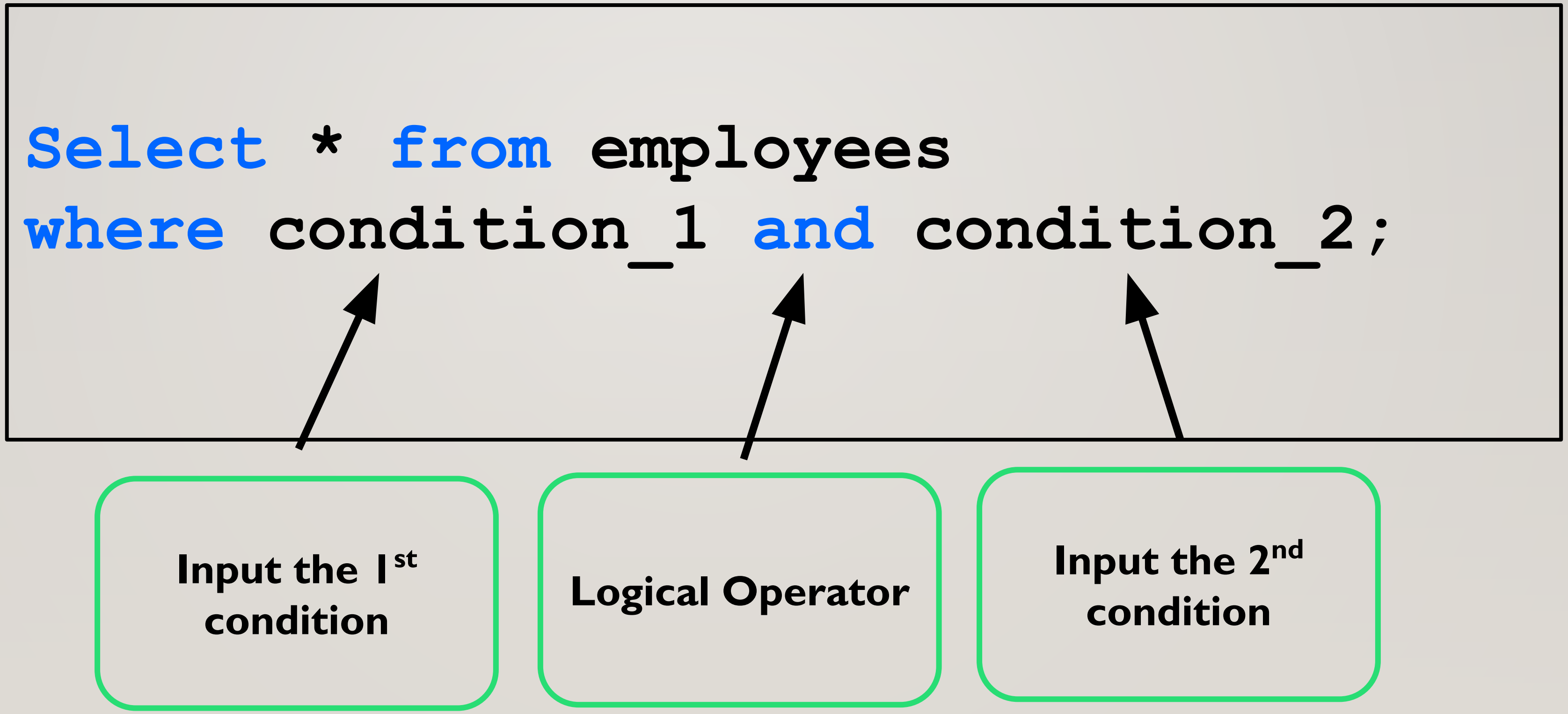
```
Select * from Table_name
where condition;
```

Input the condition

SYNTAX : SELECT

```
Select * from employees
where department_id = 50;
```

SYNTAX : SELECT



```
Select * from employees
where condition_1 and condition_2;
```

Input the 1st condition

Logical Operator

Input the 2nd condition

SYNTAX : SELECT

```
Select * from employees
where department_id = 50 and
manager_id = 20;
```

Joins In SQL

Table 1

Id	Age	Gender	Salary	City

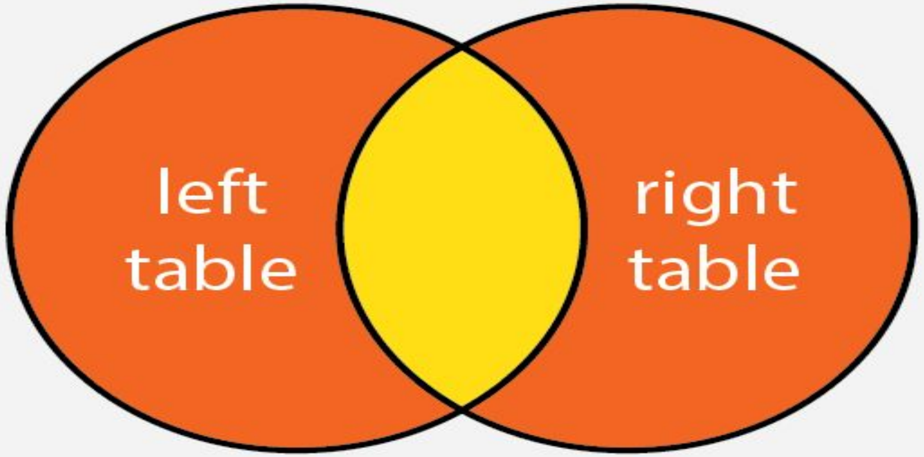
Table 2

Id	Name	Dept.	Manager

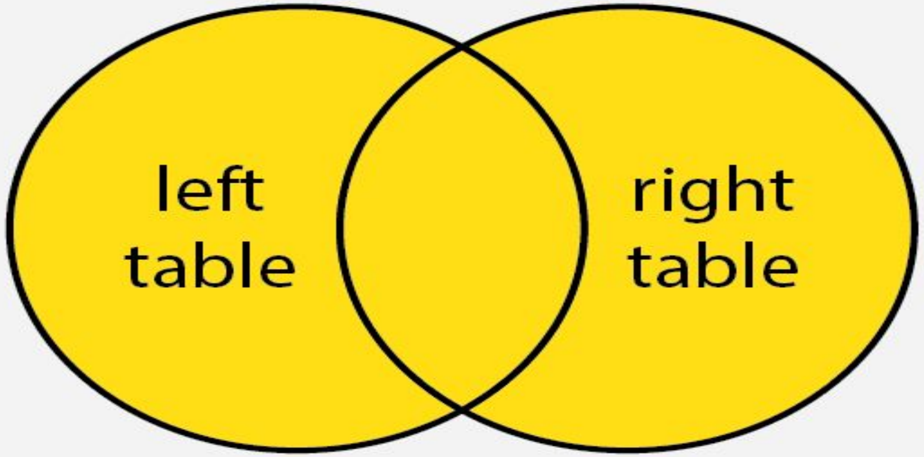
Combined Table

Id	Age	Gender	Dept.	Manager

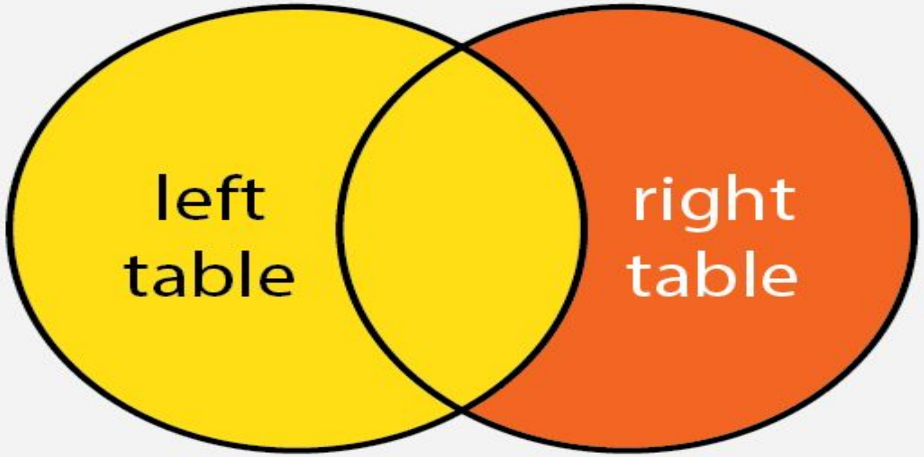
INNER JOIN



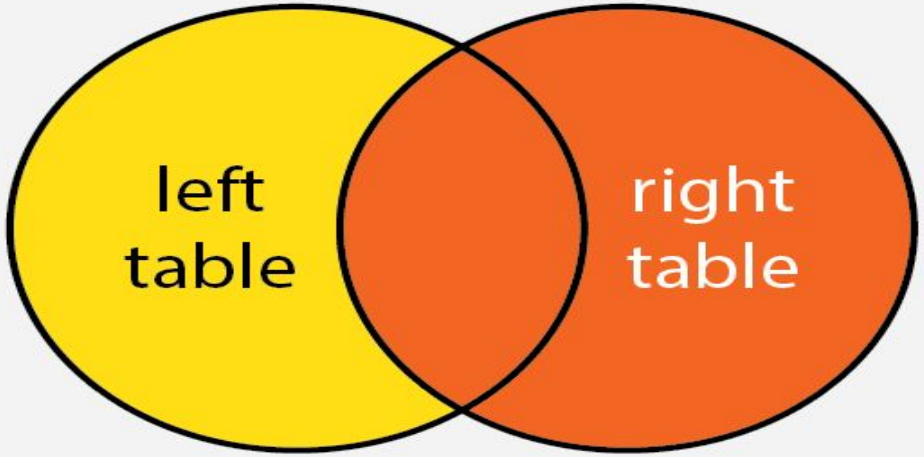
FULL JOIN



LEFT JOIN



RIGHT JOIN



INNER JOIN

Table 1

Id	Age	Gender	Salary	City
201	32	M	20K	Beng
202	33	F	25K	Mum
203	22	F	20K	Mum
204	23	M	22K	Che

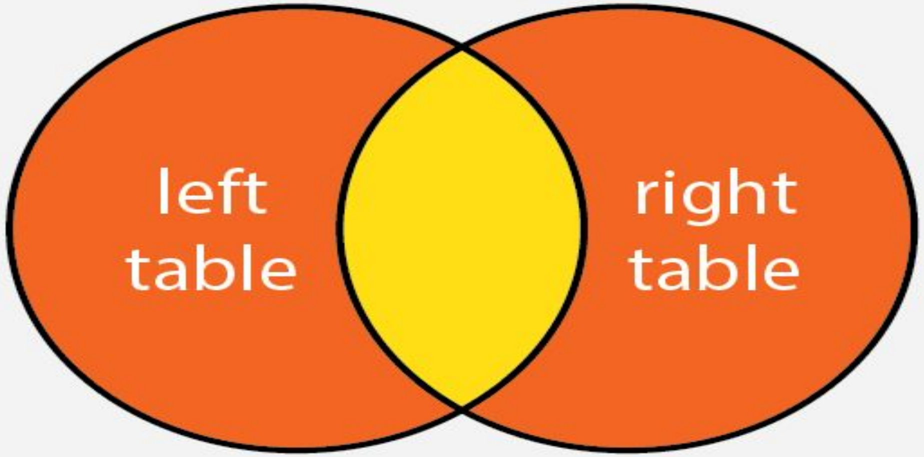
Table 2

Id	Name	Dept.	Manager
202	Shree	Mar	Ram
204	Ram	Fin	Atul
211	Priya	HR	Raj
212	Ritu	Ops	Amar

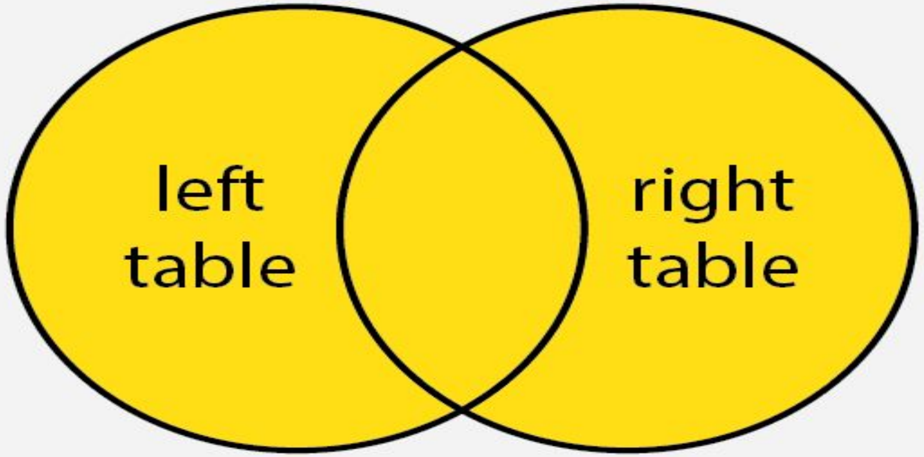
Combined Table

Id	Age	Gender	Dept.	Manager
202	**33**	**F**	**Mar**	**Ram**
204	**23**	**M**	**Fin**	**Atul**

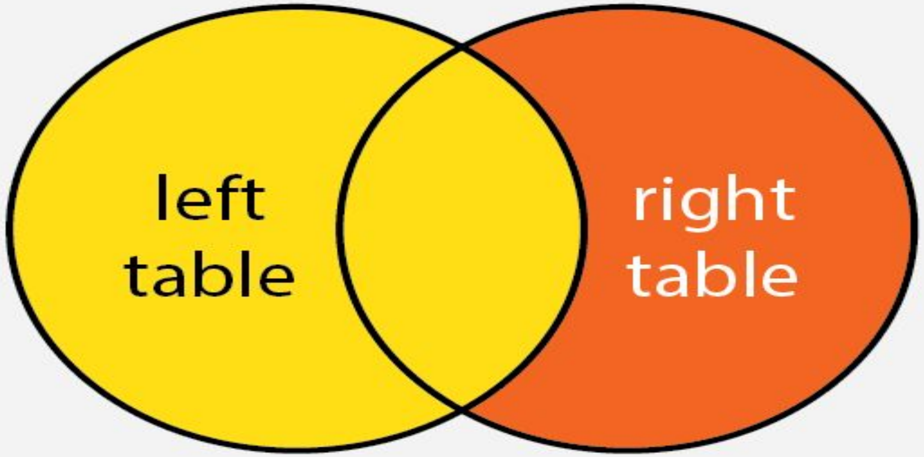
INNER JOIN



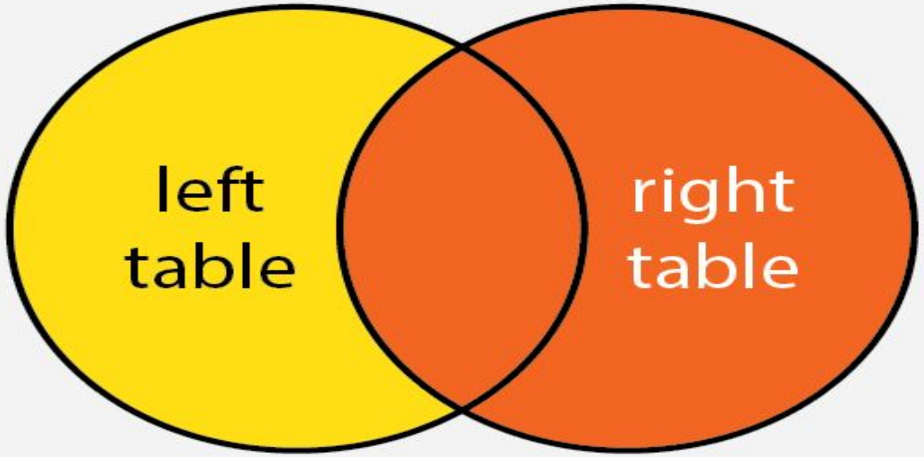
FULL JOIN



LEFT JOIN



RIGHT JOIN



FULL JOIN

Table 1

Id	Age	Gender	Salary	City
201	32	M	20K	Beng
202	33	F	25K	Mum
203	22	F	20K	Mum
204	23	M	22K	Che

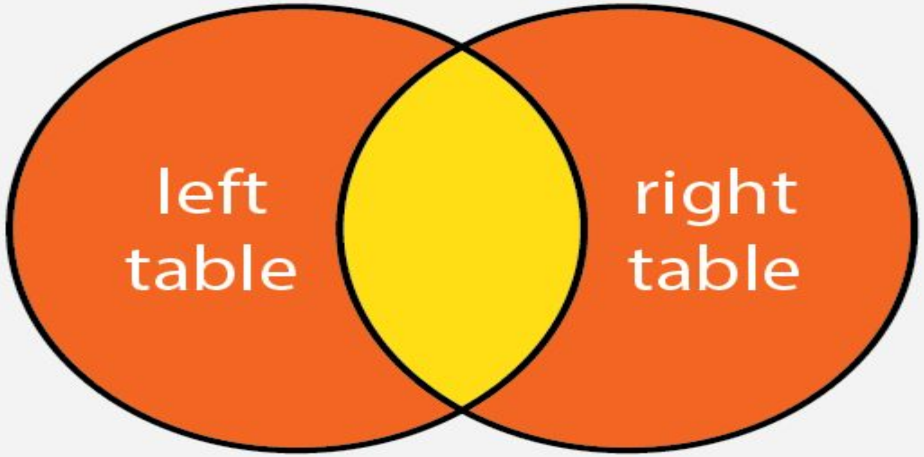
Table 2

Id	Name	Dept.	Manager
202	Shree	Mar	Ram
204	Ram	Fin	Atul
211	Priya	HR	Raj
212	Ritu	Ops	Amar

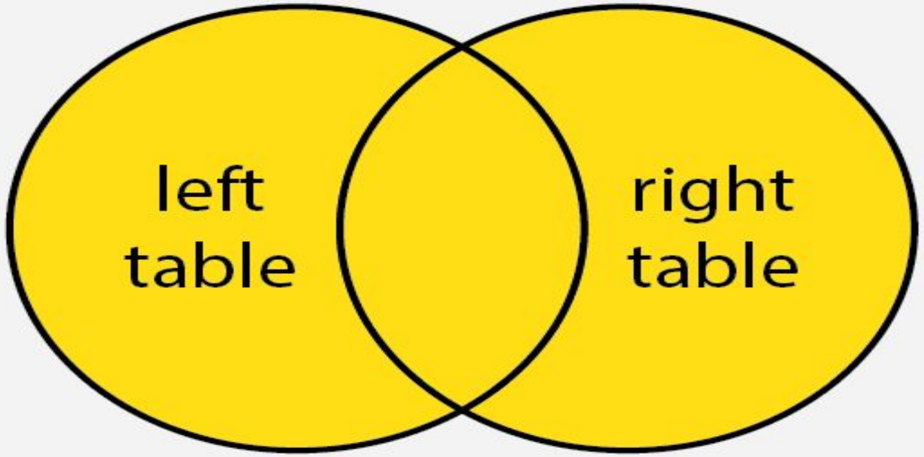
Combined Table

Id	Age	Gender	Dept.	Manager
201	32	M	Null	Null
202	33	F	Mar	Ram
203	22	F	Null	Null
204	23	M	Fin	Atul
211	Null	Null	HR	Raj
212	Null	Null	Ops	Amar

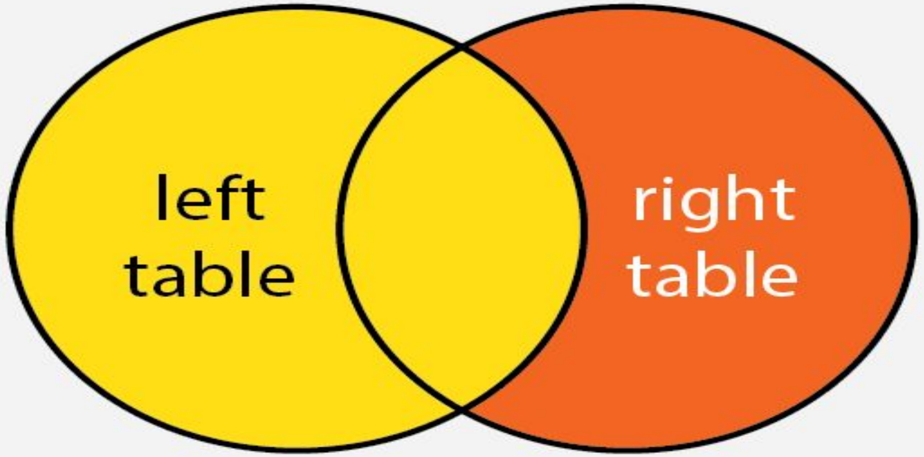
INNER JOIN



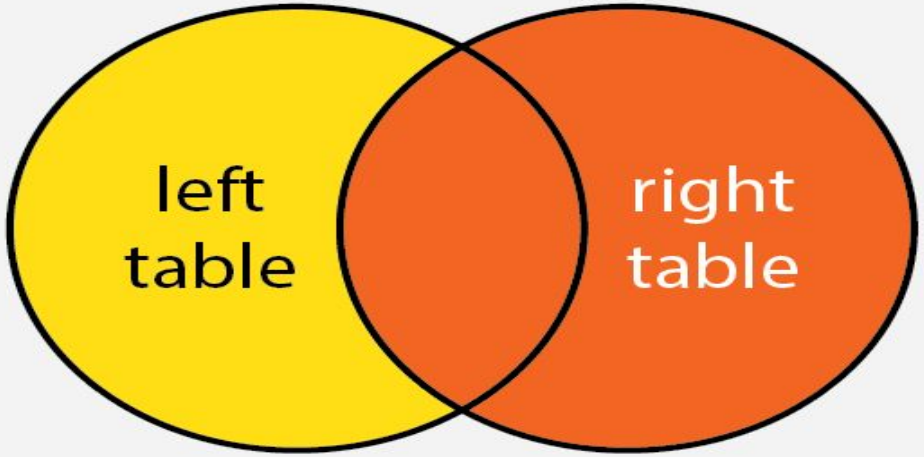
FULL JOIN



LEFT JOIN



RIGHT JOIN



LEFT JOIN

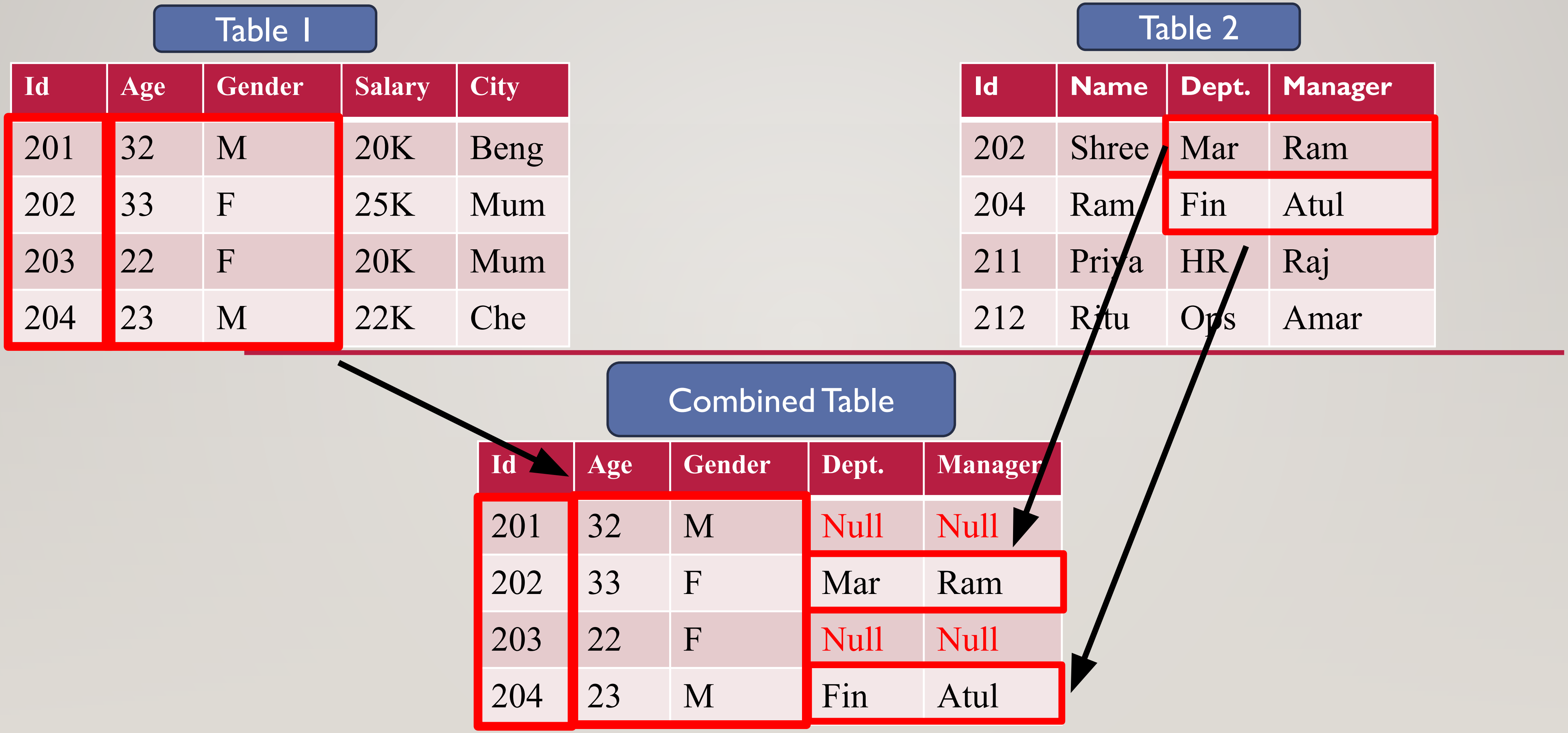


Table 1

Id	Age	Gender	Salary	City
201	32	M	20K	Beng
202	33	F	25K	Mum
203	22	F	20K	Mum
204	23	M	22K	Che

Table 2

Id	Name	Dept.	Manager
202	Shree	Mar	Ram
204	Ram	Fin	Atul
211	Priya	HR	Raj
212	Ritu	Ops	Amar

Combined Table

Id	Age	Gender	Dept.	Manager
201	32	M	Null	Null
202	33	F	Mar	Ram
203	22	F	Null	Null
204	23	M	Fin	Atul

LEFT JOIN

Table 1

Id	Age	Gender	Salary	City
201	32	M	20K	Beng
202	33	F	25K	Mum
203	22	F	20K	Mum
204	23	M	22K	Che

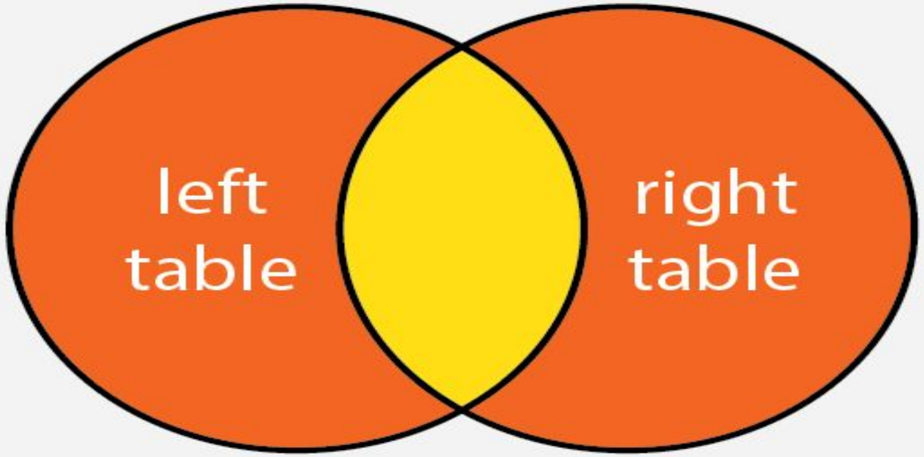
Table 2

Id	Name	Dept.	Manager
202	Shree	Mar	Ram
204	Ram	Fin	Atul
211	Priya	HR	Raj
212	Ritu	Ops	Amar

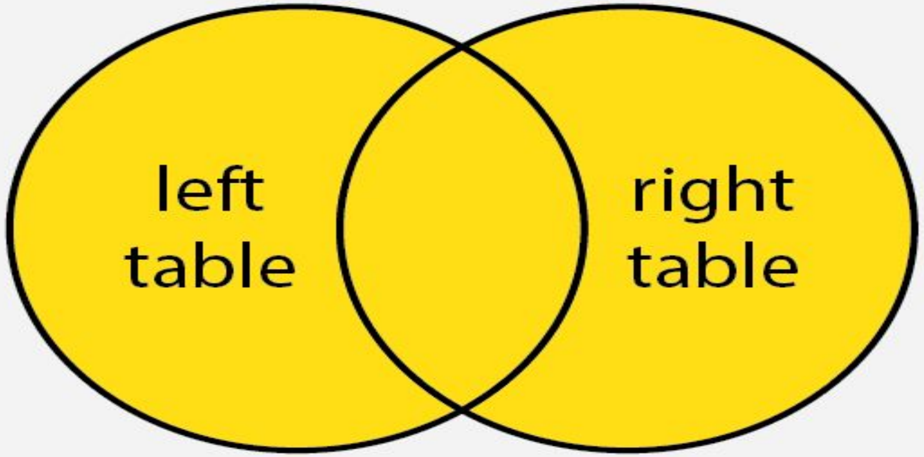
Combined Table

Id	Age	Gender	Dept.	Manager
201	32	M	Null	Null
202	33	F	Mar	Ram
203	22	F	Null	Null
204	23	M	Fin	Atul

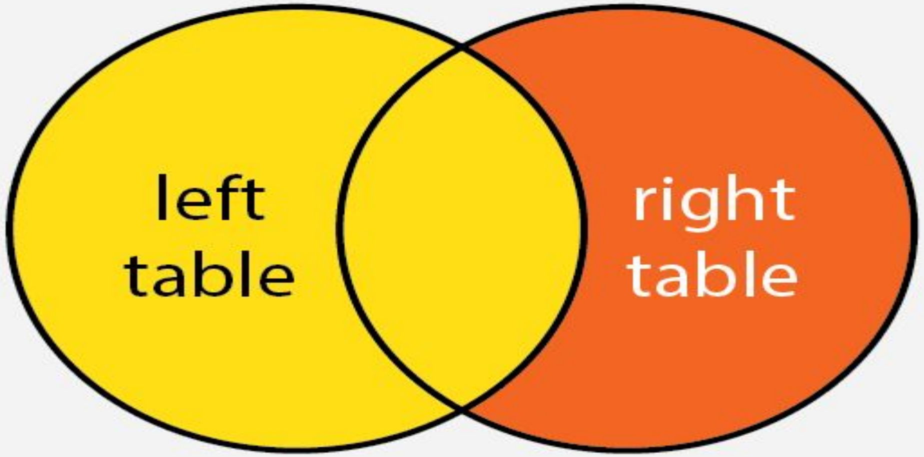
INNER JOIN



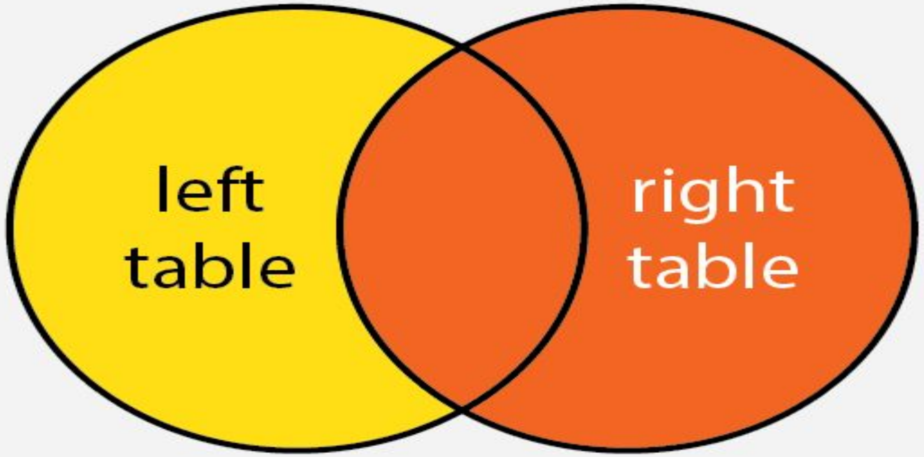
FULL JOIN



LEFT JOIN



RIGHT JOIN



RIGHT JOIN

Table 1

Id	Age	Gender	Salary	City
201	32	M	20K	Beng
202	33	F	25K	Mum
203	22	F	20K	Mum
204	23	M	22K	Che

Table 2

Id	Name	Dept.	Manager
202	Shree	Mar	Ram
204	Ram	Fin	Atul
211	Priya	HR	Raj
212	Ritu	Ops	Amar

Combined Table

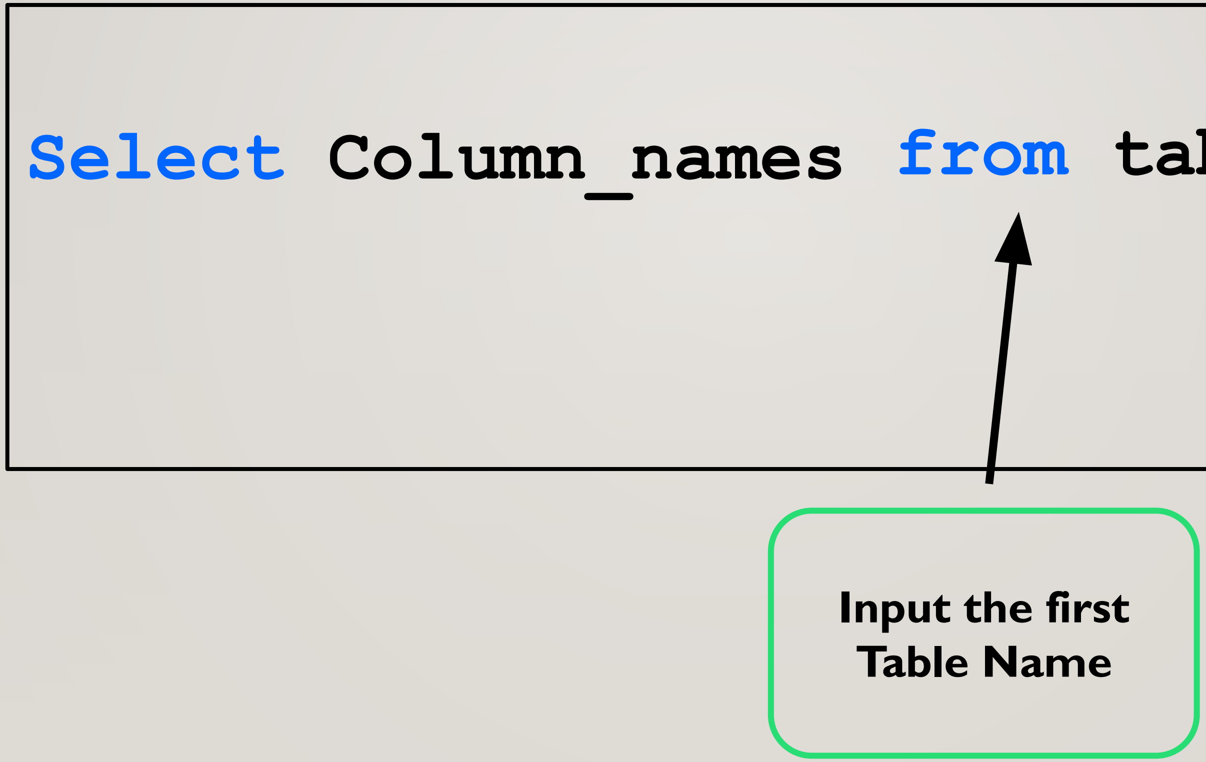
Id	Age	Gender	Dept.	Manager
202	33	F	Mar	Ram
204	23	M	Fin	Atul
211	Null	Null	HR	Raj
212	Null	Null	Ops	Amar

SYNTAX : SELECT

```
Select Column_names
```

Use Select Statement to Select Columns

SYNTAX : SELECT



```
Select Column_names from table_1
```

Input the first Table Name

SYNTAX : SELECT

```
Select Column_names from table_1
Type_of_join table_2
```

Input the type of join and other table/s

SYNTAX : SELECT

```
Select Column_names from table_1
Type_of_join table_2
On table_1.common_column =
table_2.common_column
```

Input the Common Column condition

FILTERING THE DATA : RELATIONAL OPERATORS

RELATIONAL OPERATORS

Equal to (=)
Greater Than (>)
Less Than (<)
Greater than or equal to (≥, >=)
Less than or equal to (≤, <=)
Not Equal to (<>)

SYNTAX : SELECT

```
Select * from payments
where amount = 20000;
```

SYNTAX : SELECT

```
Select * from payments
where amount <= 40000;
```

SYNTAX : SELECT

```
Select * from payments
where amount > 20000;
```

SYNTAX : SELECT

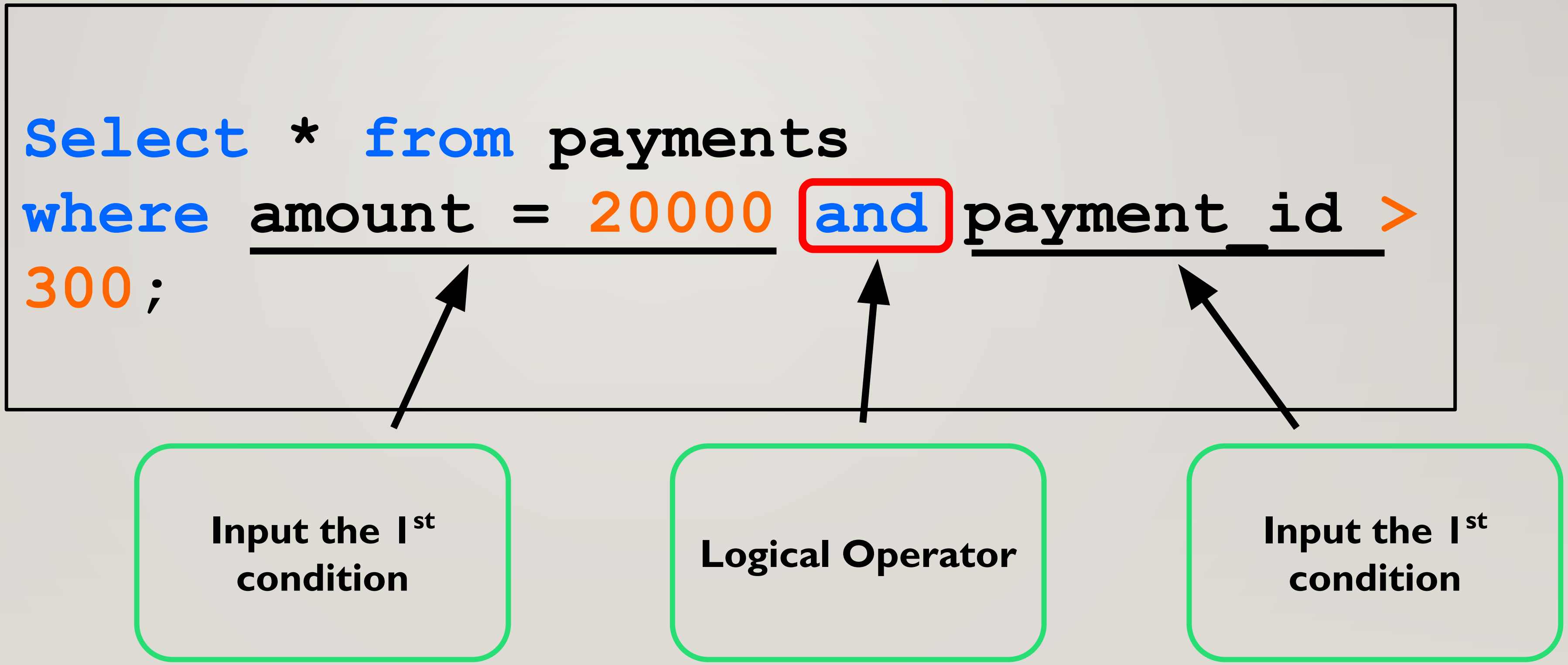
```
Select * from payments
where amount <> 20000;
```

FILTERING THE DATA : Logical Operators

LOGICAL OPERATORS : SYNTAX

```
AND

OR

NOT
```

SYNTAX : SELECT



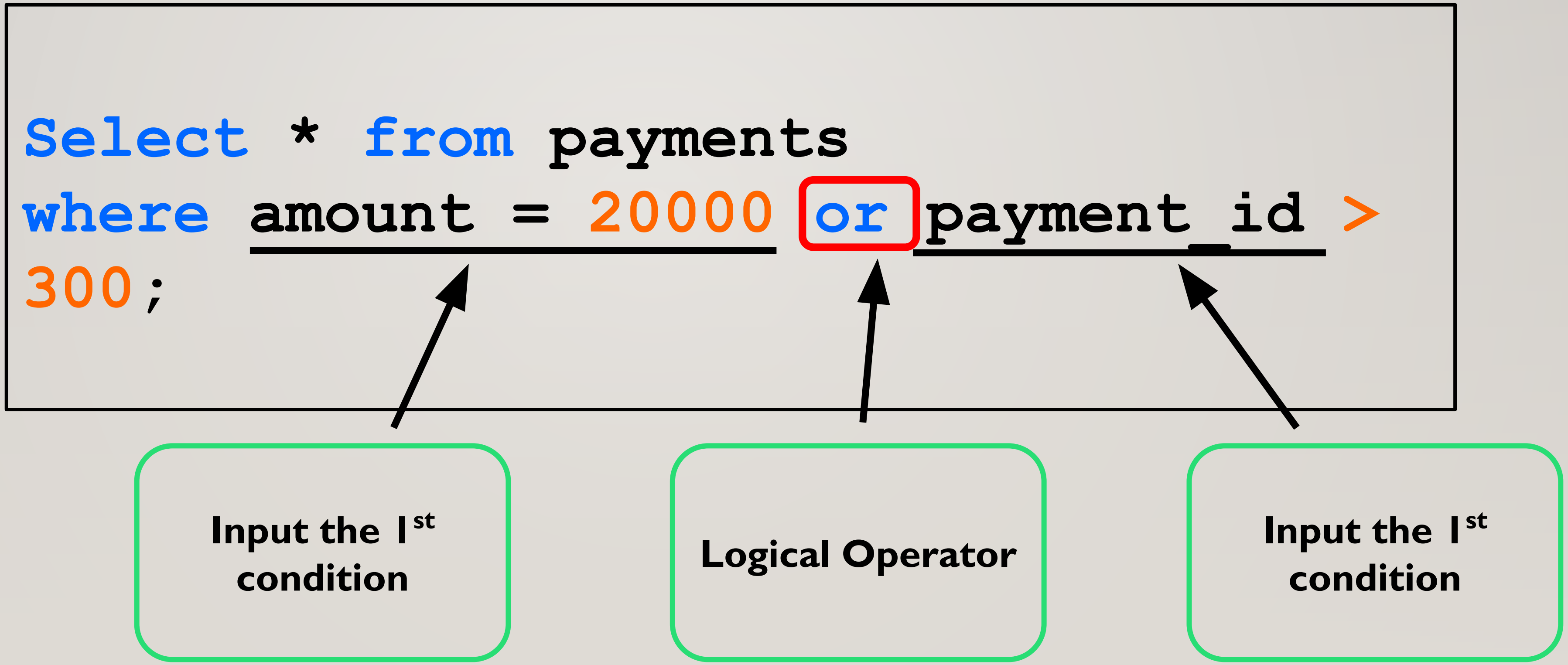
```
Select * from payments
where amount = 20000 and payment_id >
300;
```

Input the 1st condition

Logical Operator

Input the 1st condition

SYNTAX : SELECT

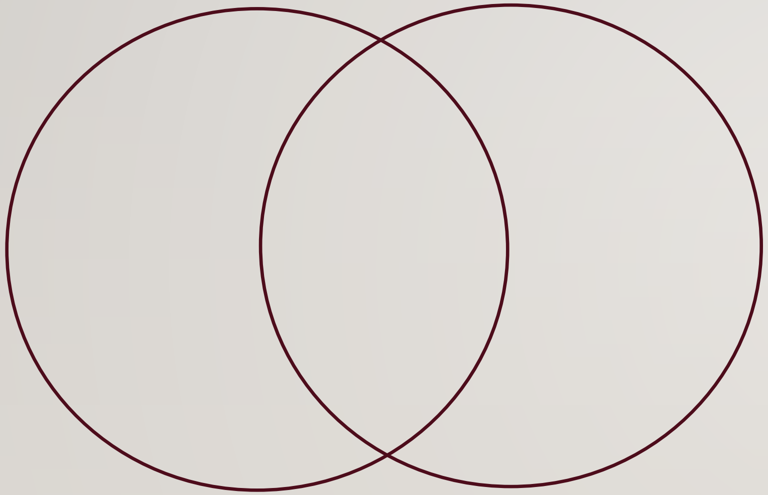


```
Select * from payments
where amount = 20000 or payment_id >
300;
```

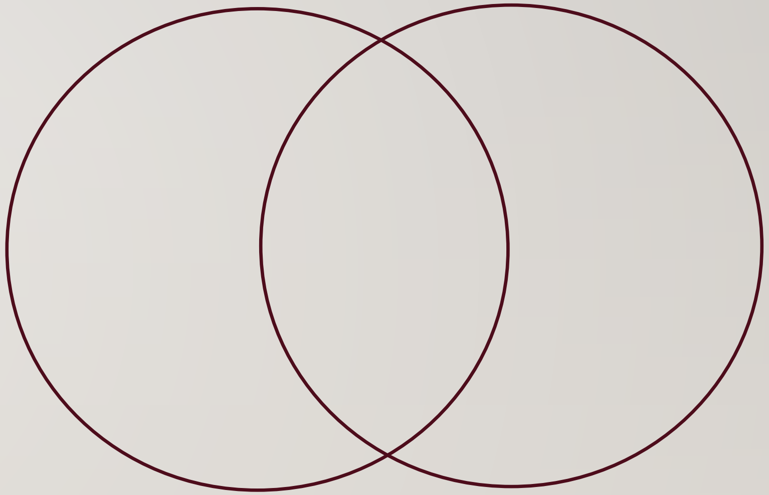
Input the 1st condition

Logical Operator

Input the 1st condition



AND



OR

SYNTAX : SELECT

```
Select * from payments
where not amount = 20000;
```

Use the NOT Operator

SYNTAX : SELECT

```
Select * from payments
where not amount = 20000 and not
customer_id > 3;
```

FILTERING THE DATA :
LIKE, IN AND BETWEEN OPERATORS

LIKE OPERATORS : SYNTAX

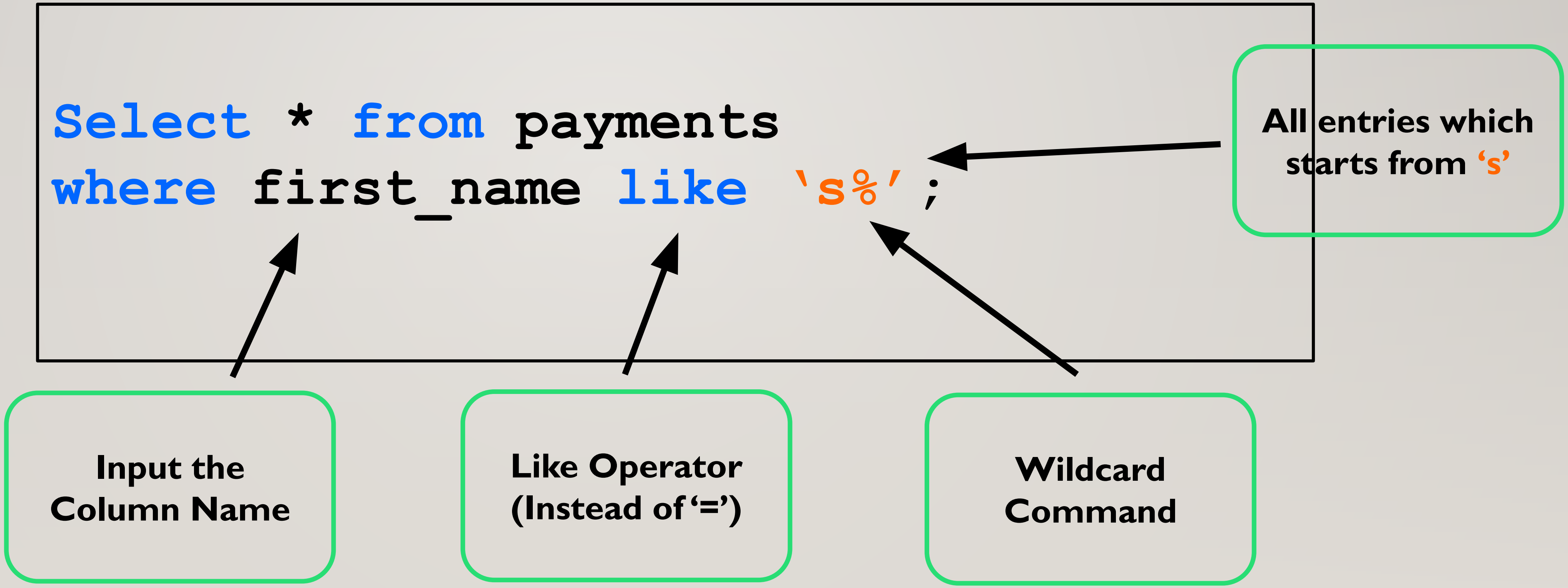
```
LIKE

IN

BETWEEN
```

LIKE OPERATOR : SYNTAX

% - Any Number of Characters

_ - Fixed Number of Characters

SYNTAX : LIKE



```
Select * from payments
where first_name like ‘s%’;
```

All entries which starts from ‘s’

Input the Column Name

Like Operator (Instead of ‘=’)

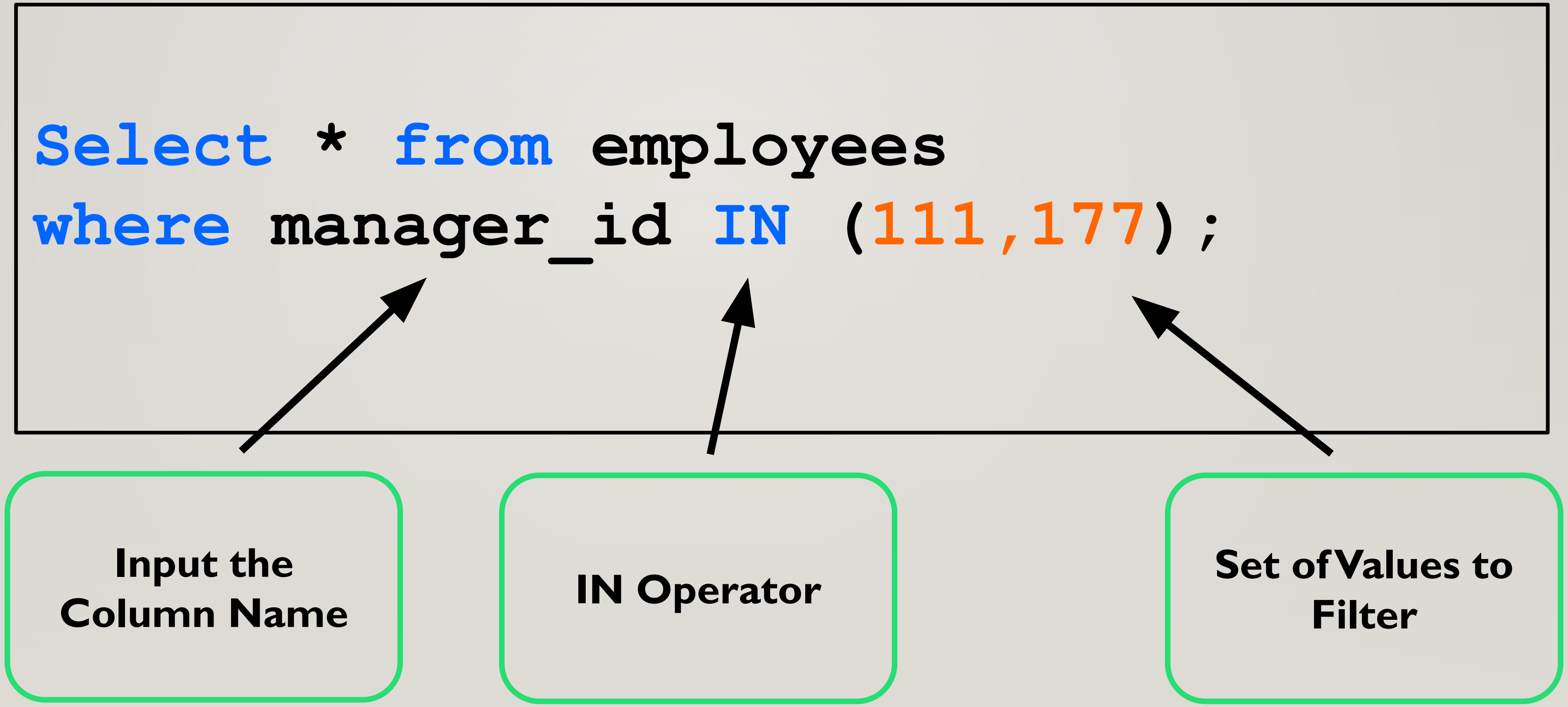
Wildcard Command

SYNTAX : LIKE

```
Select * from payments
where first_name like ‘%s’;
```

All entries which ends with ‘s’

SYNTAX : IN



```
Select * from employees
where manager_id IN (111,177);
```

Input the Column Name

IN Operator

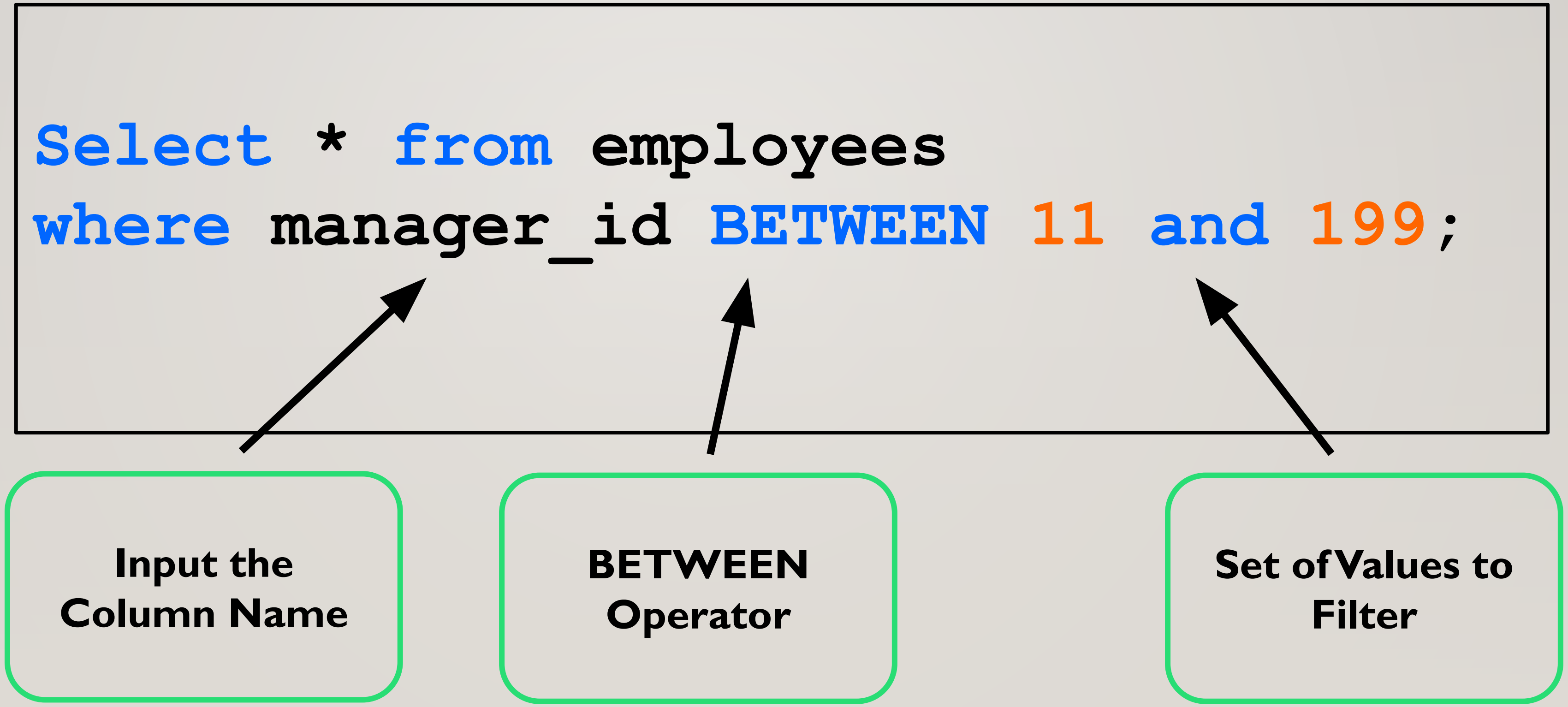
Set of Values to Filter

SYNTAX : IN

```
Select * from employees
where first_name IN (‘Aman’,’Candy’);
```

Text in ‘ ’

SYNTAX : BETWEEN



```
Select * from employees
where manager_id BETWEEN 11 and 199;
```

Input the Column Name

BETWEEN Operator

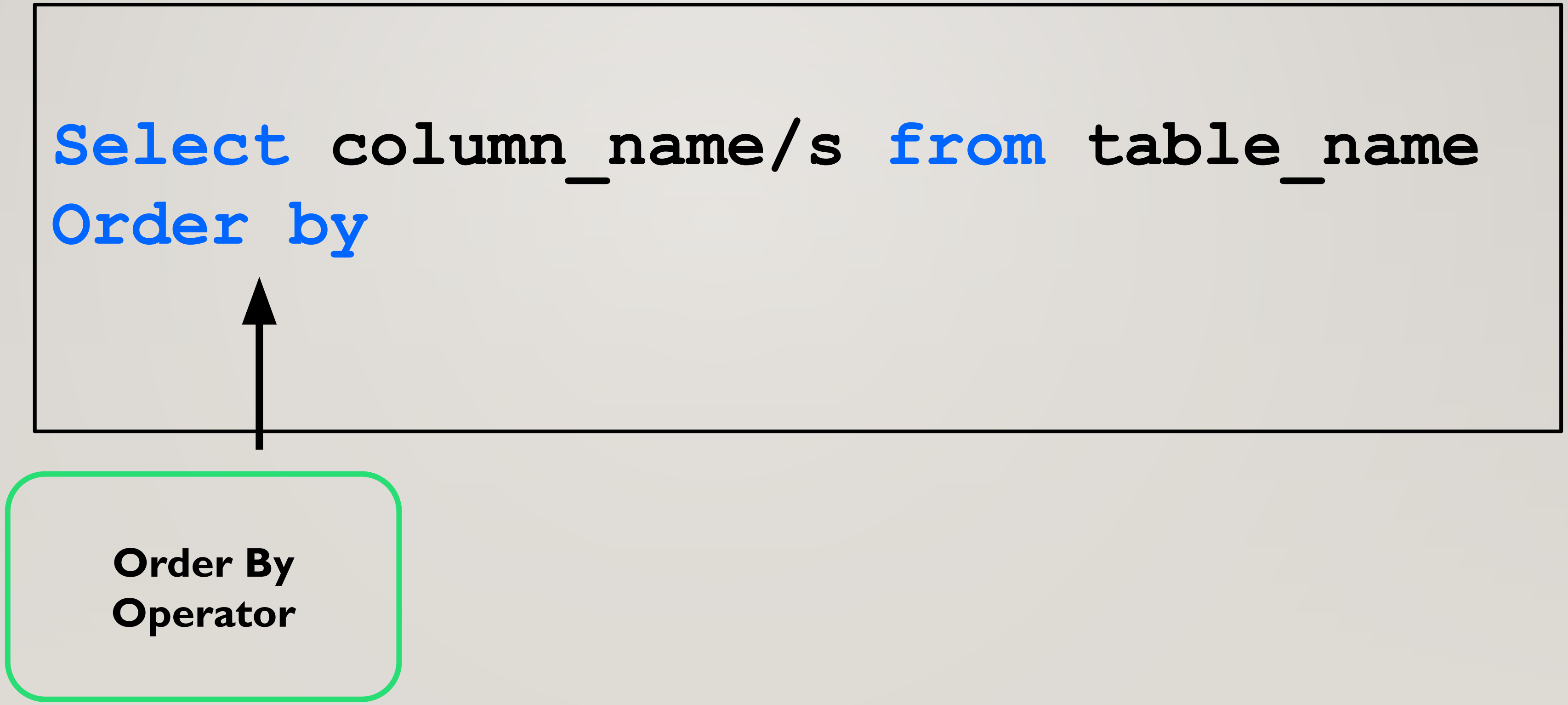
Set of Values to Filter

SYNTAX : BETWEEN

```
Select * from employees
where first_name BETWEEN ‘Brec’ and
’Gopinath’;
```

Text in ‘ ’

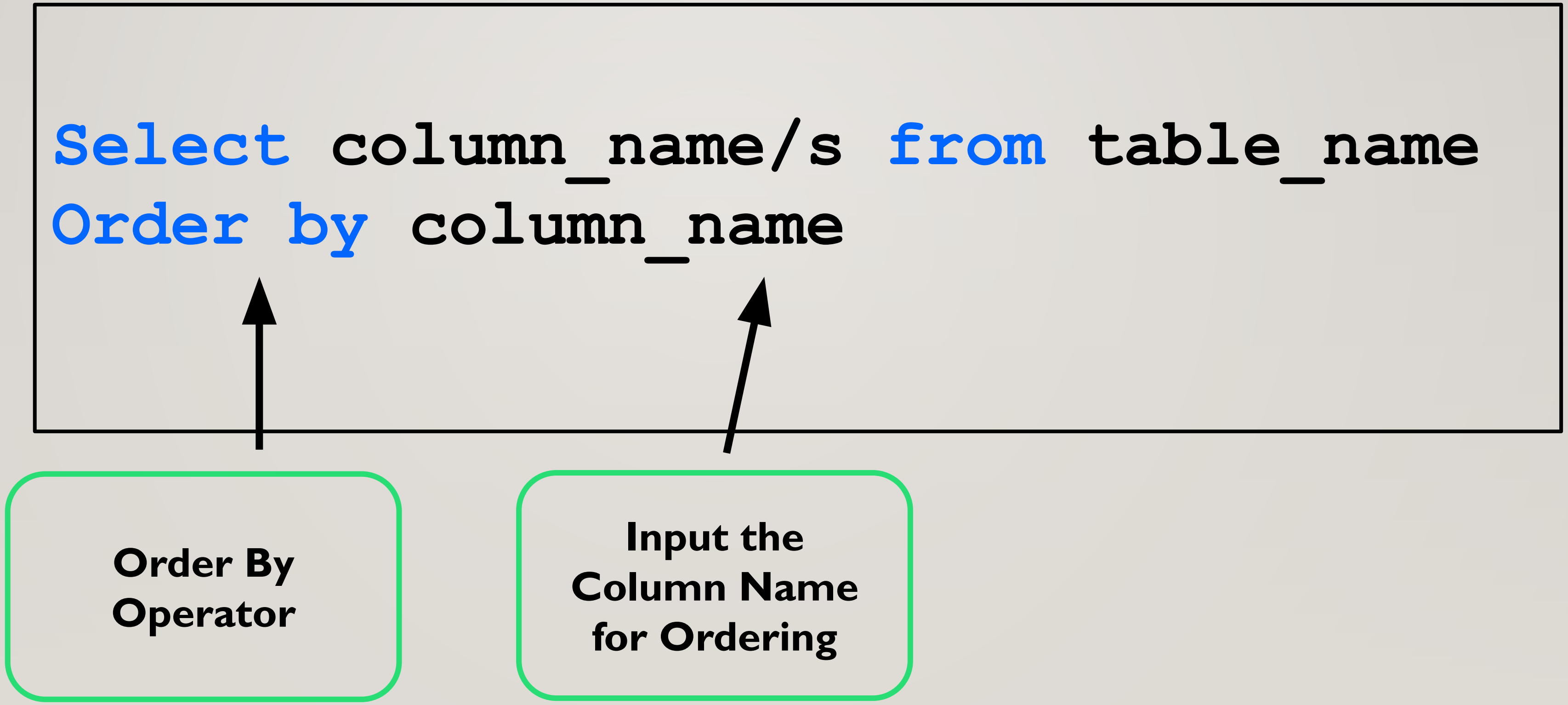
SYNTAX : ORDER BY



```
Select column_name/s from table_name
Order by
```

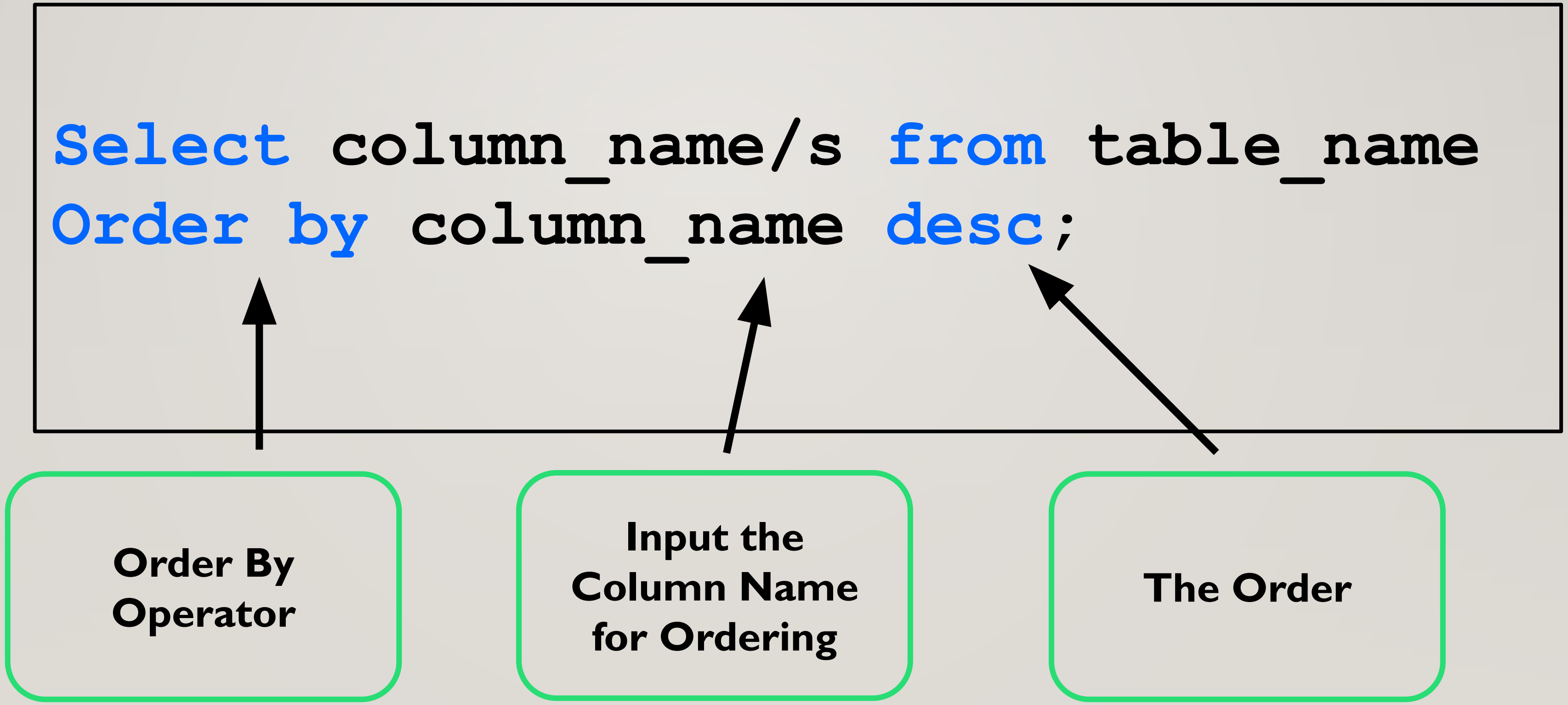
Order By Operator

SYNTAX : ORDER BY



```
Select column_name/s from table_name
Order by column_name
```

SYNTAX : ORDER BY



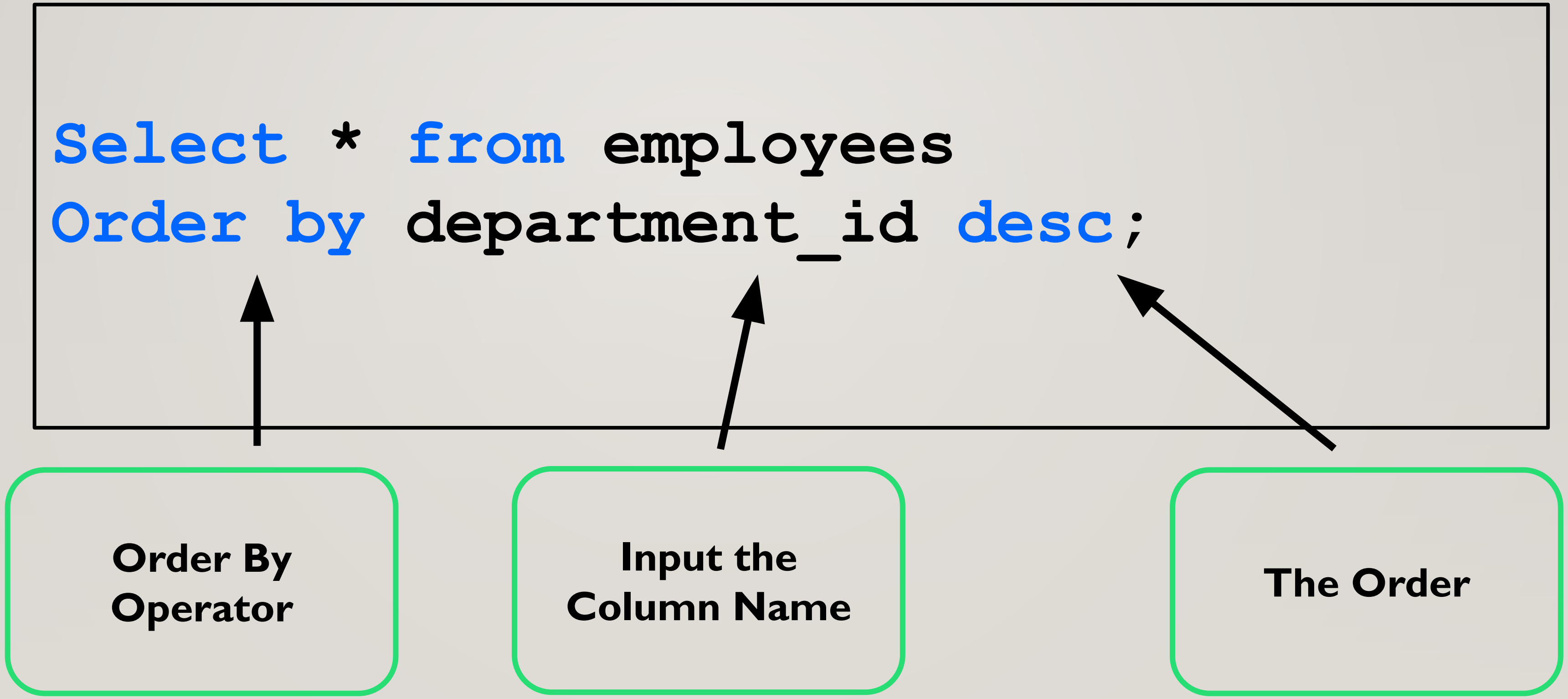
```
Select column_name/s from table_name
Order by column_name desc;
```

Order By Operator

Input the Column Name for Ordering

The Order

SYNTAX : ORDER BY



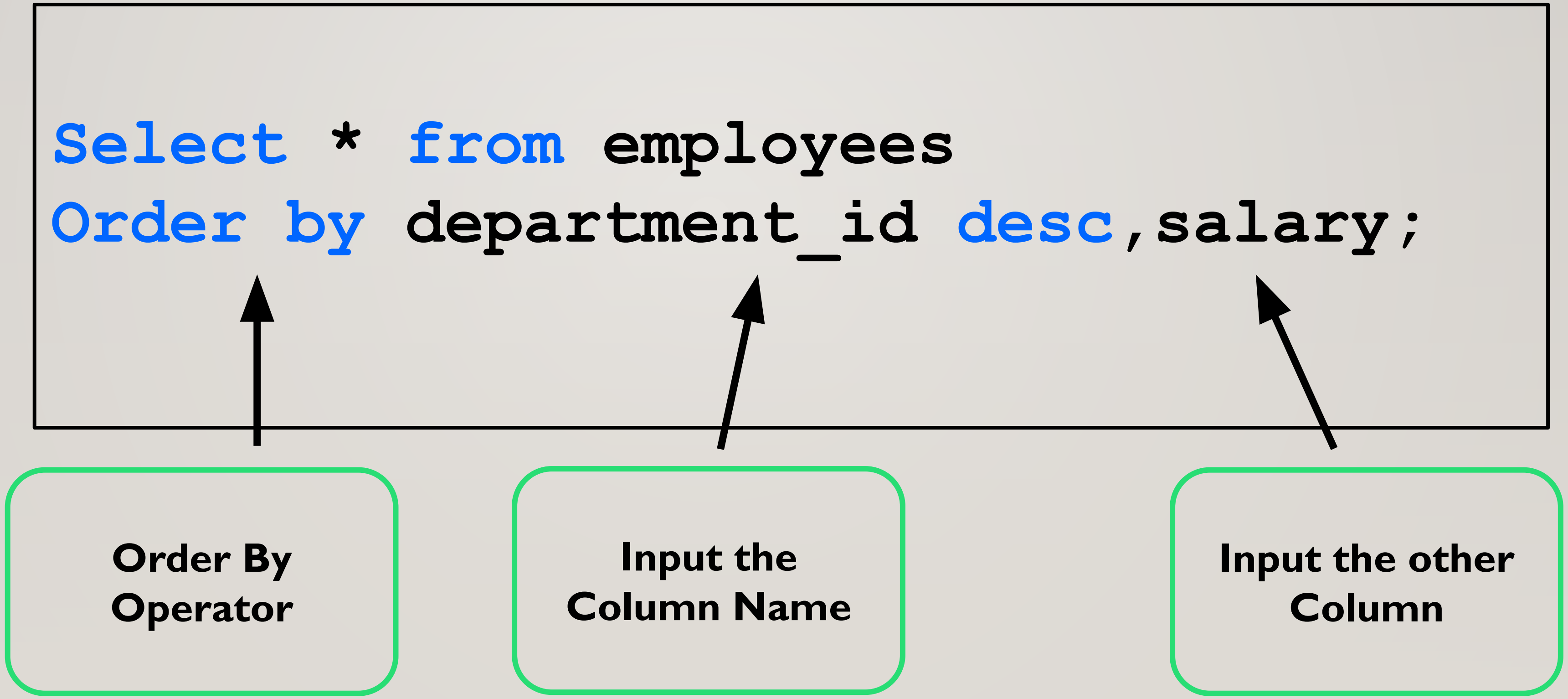
```
Select * from employees
Order by department_id desc;
```

Order By Operator

Input the Column Name

The Order

SYNTAX : ORDER BY



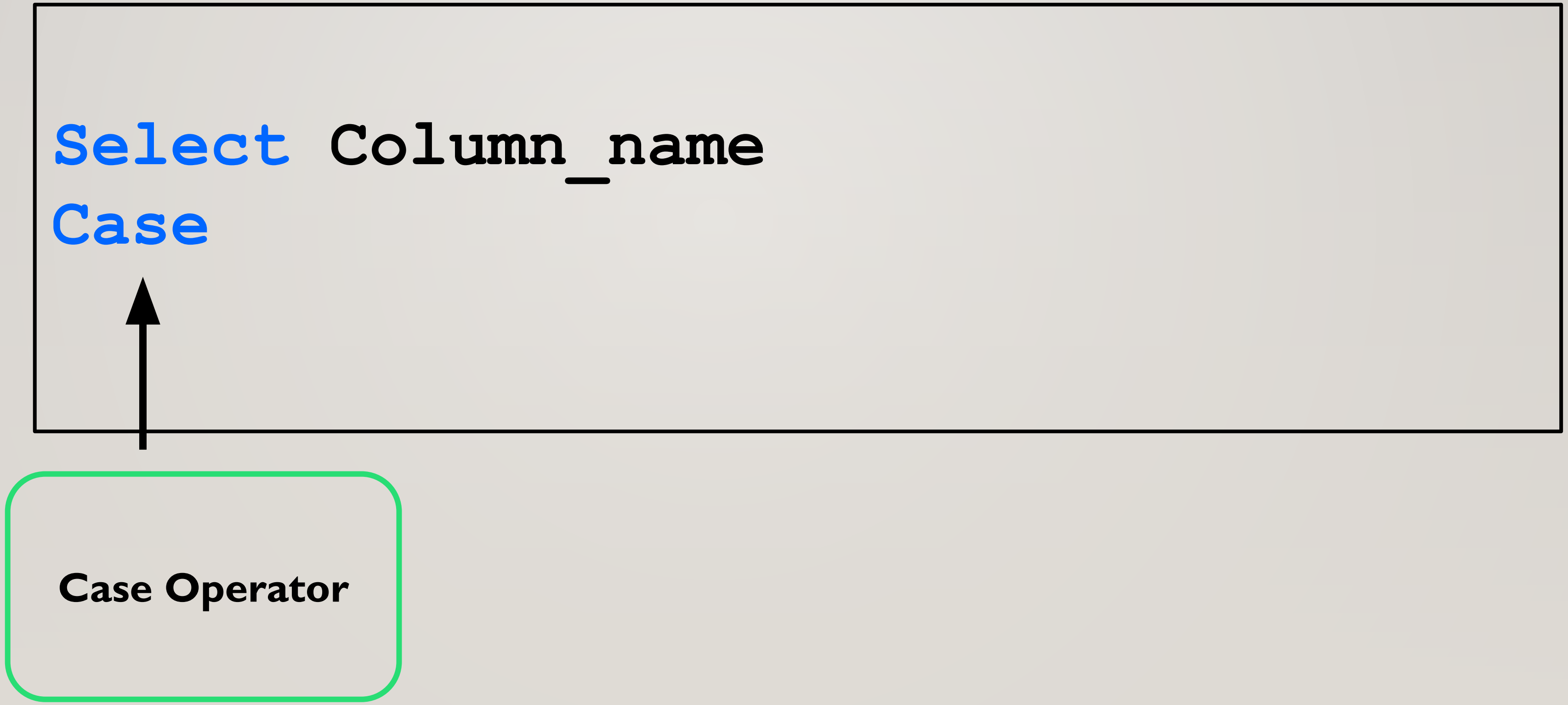
```
Select * from employees
Order by department_id desc,salary;
```

Order By Operator

Input the Column Name

Input the other Column

SYNTAX : CASE WHEN



```
Select Column_name
Case
```

Case Operator

SYNTAX : CASE WHEN

```
Select Column_names,
Case
When Condition_1
```

Input the Condition

SYNTAX : CASE WHEN

```
Select Column_names,
Case
When Condition_1 then Output_1
```

Input the Output

SYNTAX : CASE WHEN

```
Select Column_names,
Case
When Condition_1 then Output_1
When Condition_2 then Output_2
                            …

                           …

End
from table_name
```

Input “End” to End the logic

Complete the Syntax

SYNTAX : CASE WHEN

```
Select Column_names,
Case
When department_id = 22 then
```

Input then

SYNTAX : CASE WHEN

```
Select Column_names,
Case
When department_id = 22 then 'Marketing'
```

Input the desired Outcome after the condition

SYNTAX : CASE WHEN

```
Select Column_names,
Case
When department_id = 22 then ‘Marketing’
When department_id = 25 then ‘Sales’
```

Multiple Conditions Can be Provided

SYNTAX : CASE WHEN

```
Select Column_names,
Case
When department_id = 22 then 'Marketing'
When department_id = 25 then 'Sales'
                             …
                            …
End
```

Input "End" to End the logic

SYNTAX : CASE WHEN

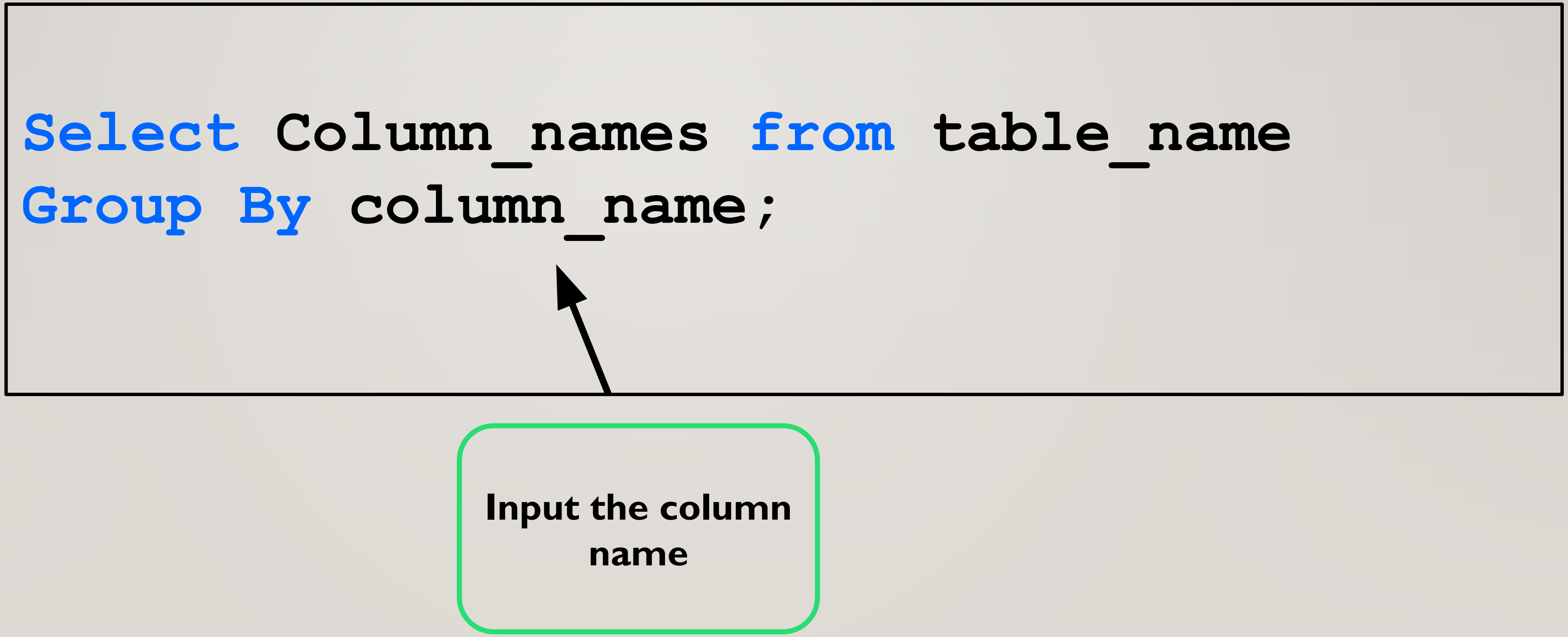
```
Select Column_names,
Case
When department_id = 22 then ‘Marketing’
When department_id = 25 then ‘Sales’
                              …

                             …

End
From employees
```

SYNTAX : GROUP BY

```
Select Column_names from table_name
Group By
```

Input Group By

SYNTAX : GROUP BY



```
Select Column_names from table_name
Group By column_name;
```

Input the column name

SYNTAX : GROUP BY

```
Select Customer_id,Product_id from
table_name
Group By Customer_id;
```

Input the column name

SYNTAX : HAVING

```
Select Column_names from table_name
Group By column_name
having condition ;
```

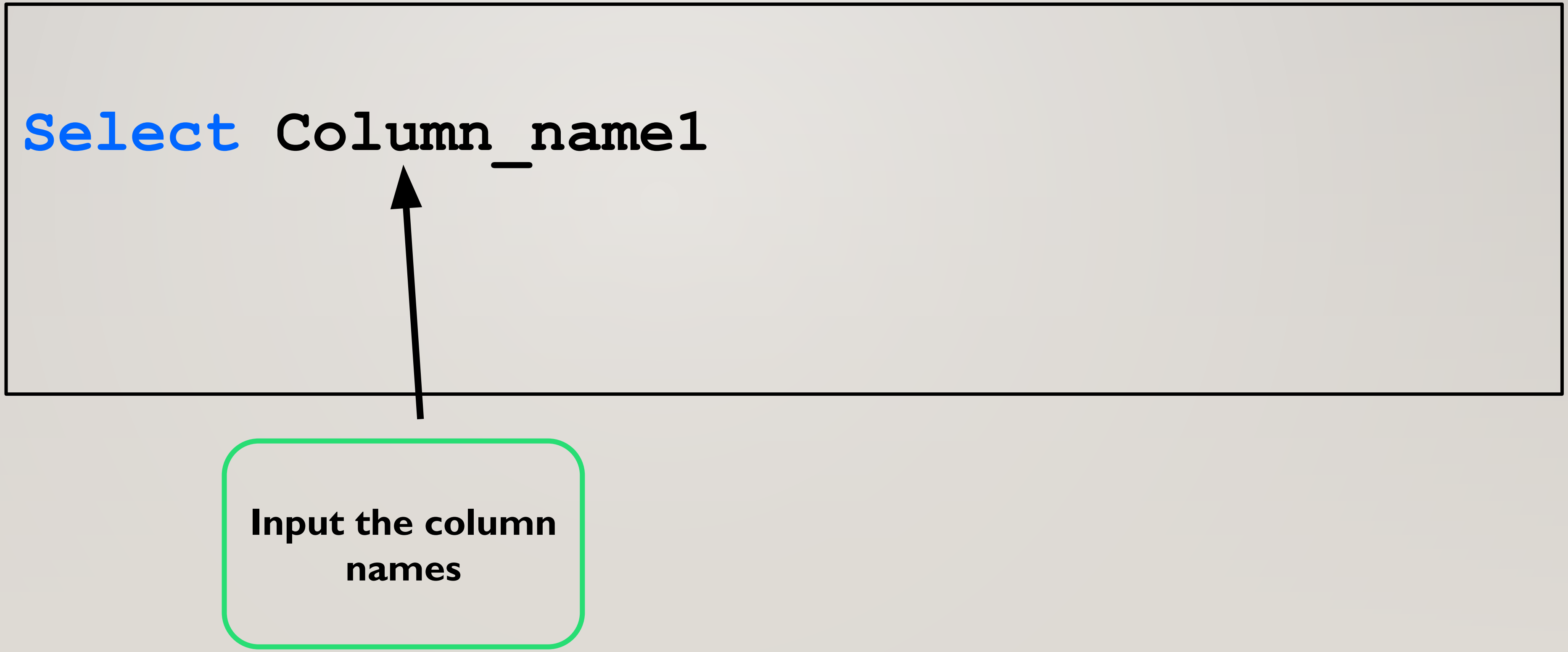
Input the having clause

SYNTAX : HAVING

```
Select Customer_id,Product_id from
table_name
Group By Customer_id
Having Product_id > 3;
```

Input the Condition

SYNTAX :ALIASES



```
Select Column_name1
```

Input the column names

SYNTAX :ALIASES

```
Select Column_name1 as a1
```

Input the alias

SYNTAX :ALIASES

```
Select Column_name1 as a1,Column_name 2 as a2
```

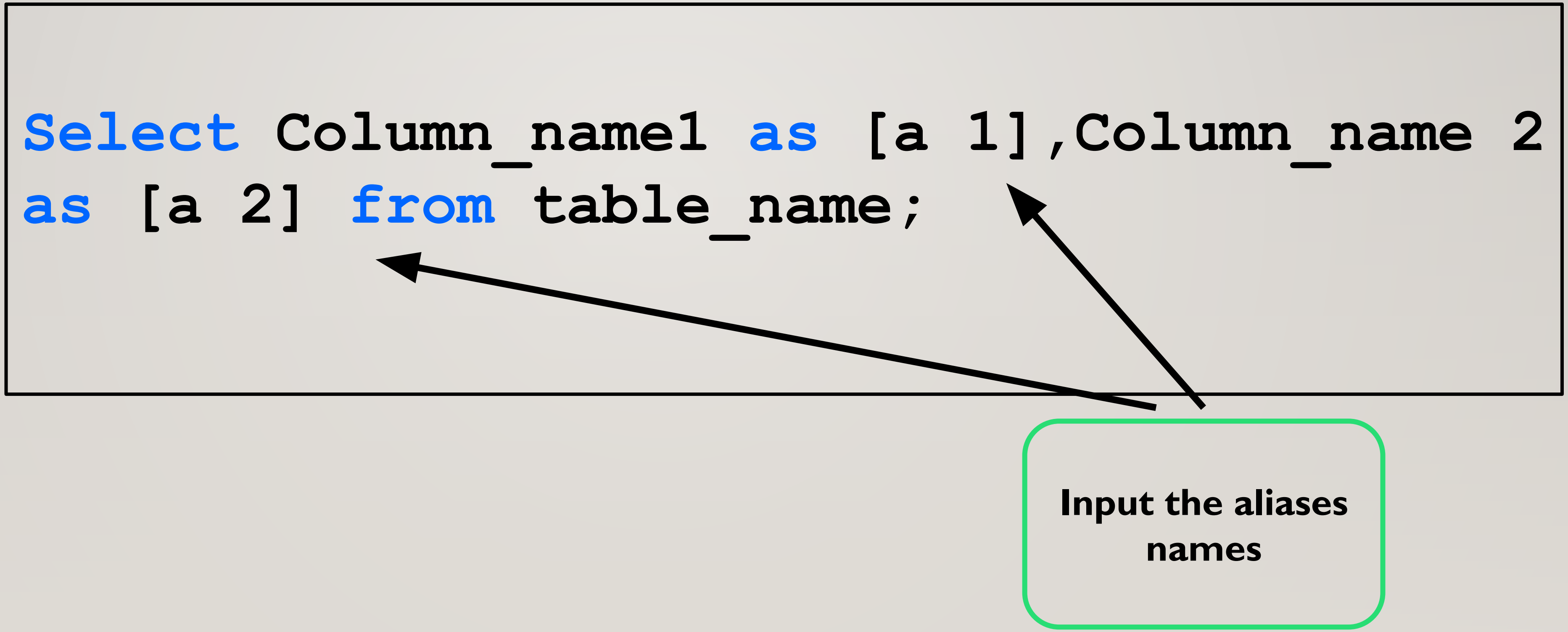
Input all column names and aliases

SYNTAX :ALIASES

```
Select Column_name1 a1,Column_name 2 a2
from table_name;
```

Complete the code

SYNTAX :ALIASES



```
Select Column_name1 as [a 1],Column_name 2
as [a 2] from table_name;
```

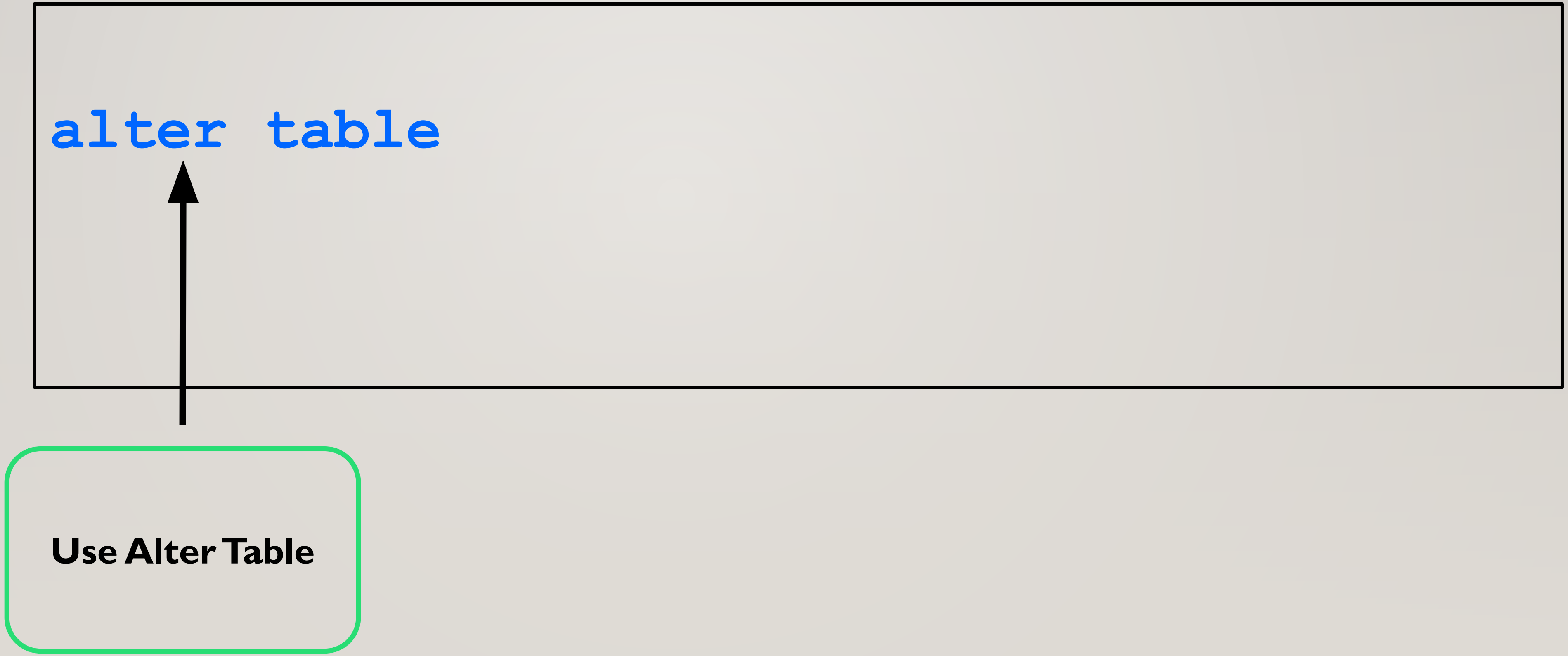
Input the aliases names

SYNTAX :ALIASES

```
Select Column_name1 a1,Column_name 2 a2
from table_name;
```

Complete the code

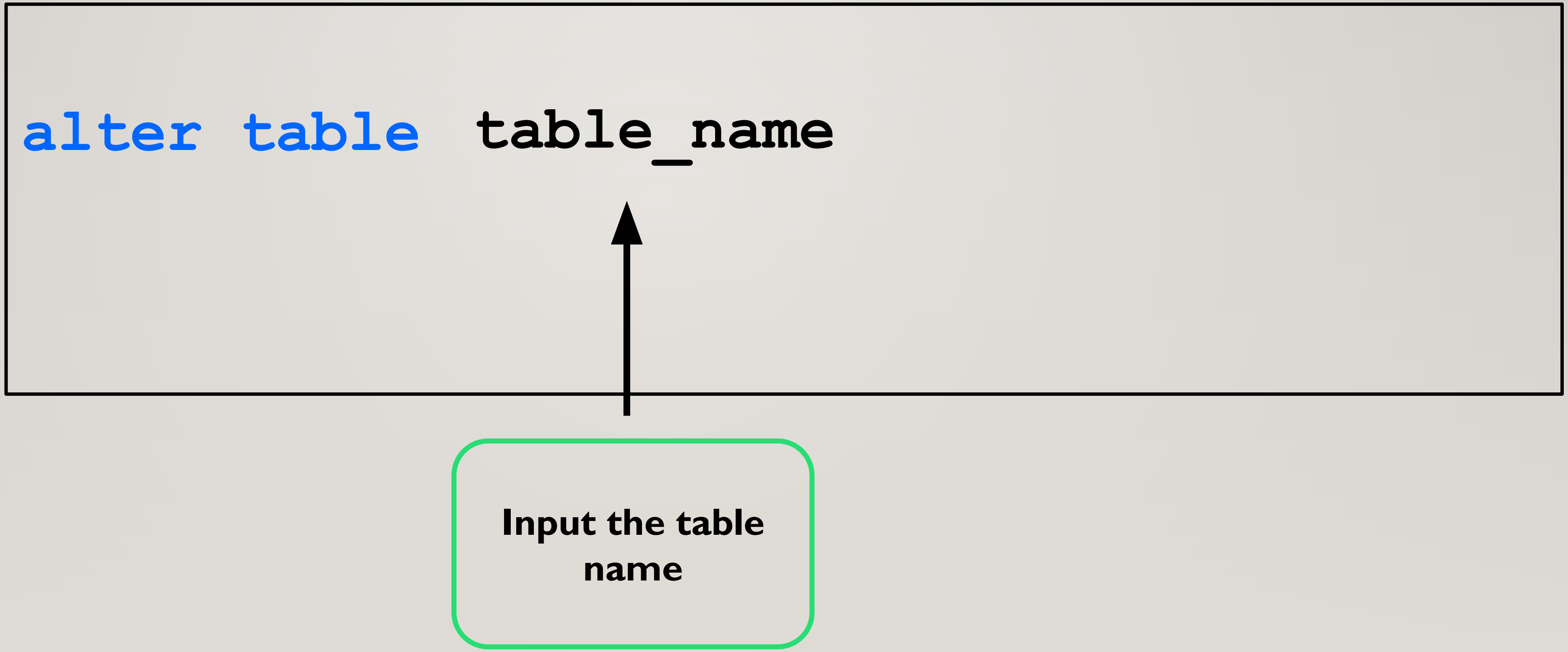
SYNTAX : ALTERING TABLE



```
alter table
```

Use Alter Table

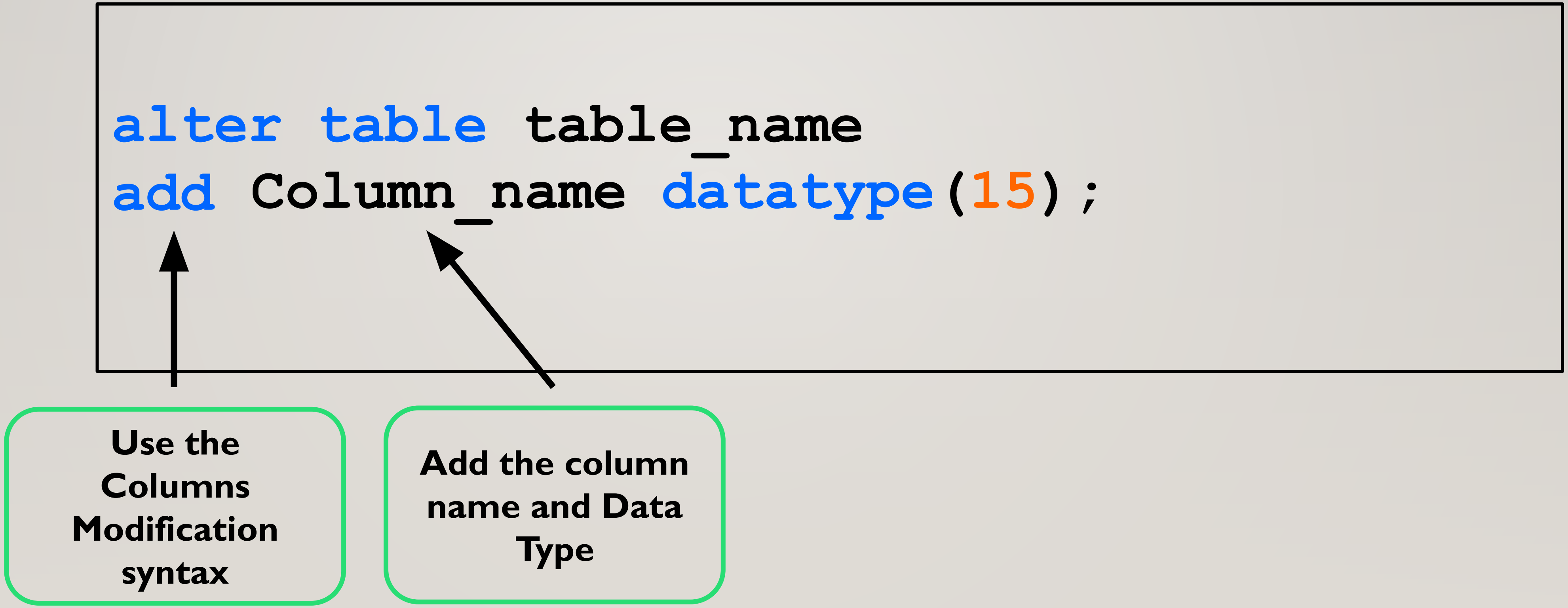
SYNTAX : ALTERING TABLE



```
alter table  table_name
```

Input the table name

SYNTAX : ALTERING TABLE



```
alter table table_name
add Column_name datatype(15);
```

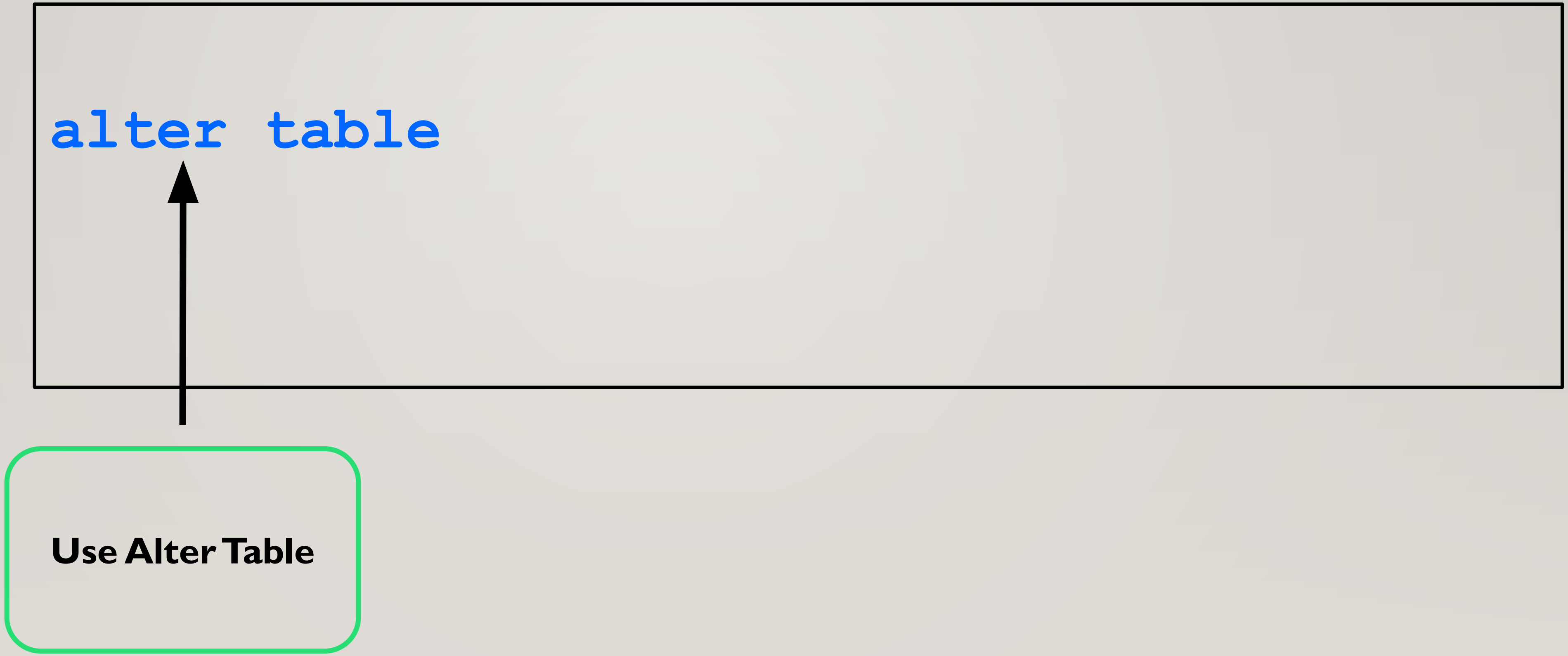
Use the Columns Modification syntax

Add the column name and Data Type

SYNTAX :ALTERING TABLE

```
alter table employees
add location Varchar(15);
```

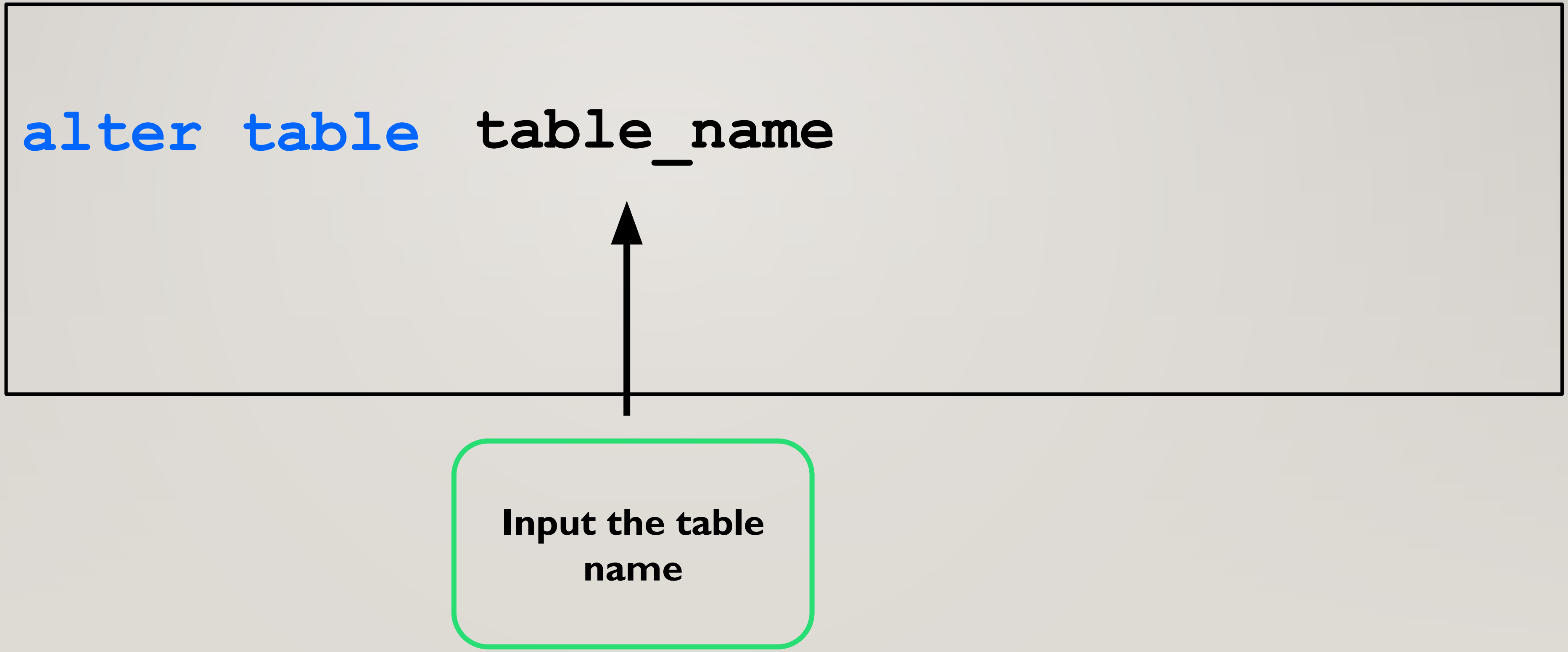
SYNTAX : ALTERING TABLE



```
alter table
```

Use Alter Table

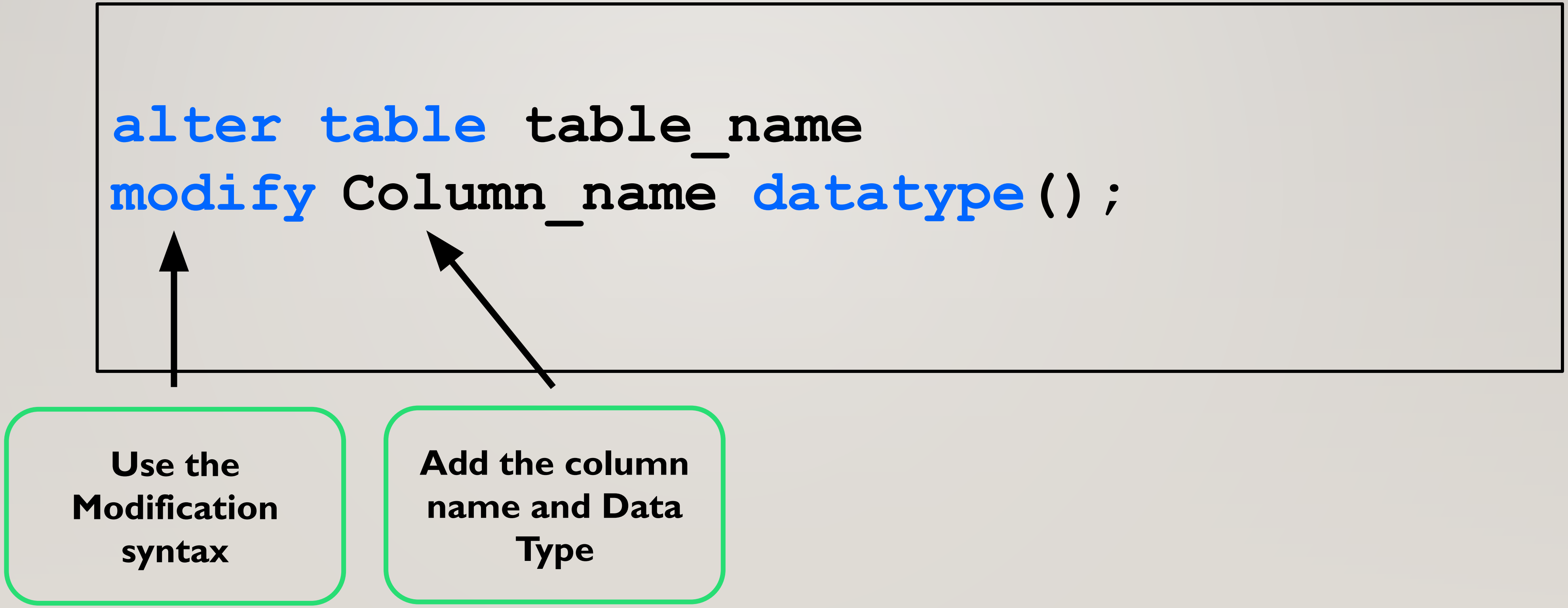
SYNTAX : ALTERING TABLE



```
alter table  table_name
```

Input the table name

SYNTAX :ALTERING TABLE

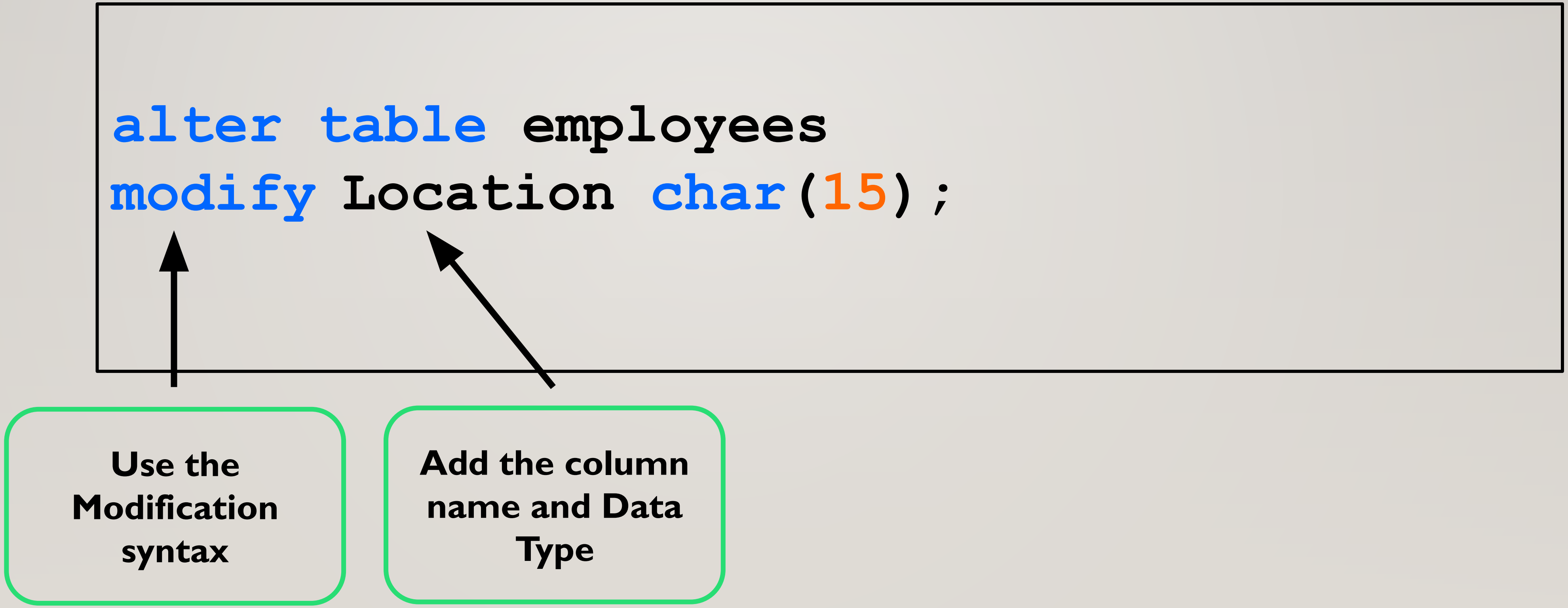


```
alter table table_name
modify Column_name datatype();
```

Use the Modification syntax

Add the column name and Data Type

SYNTAX : ALTERING TABLE

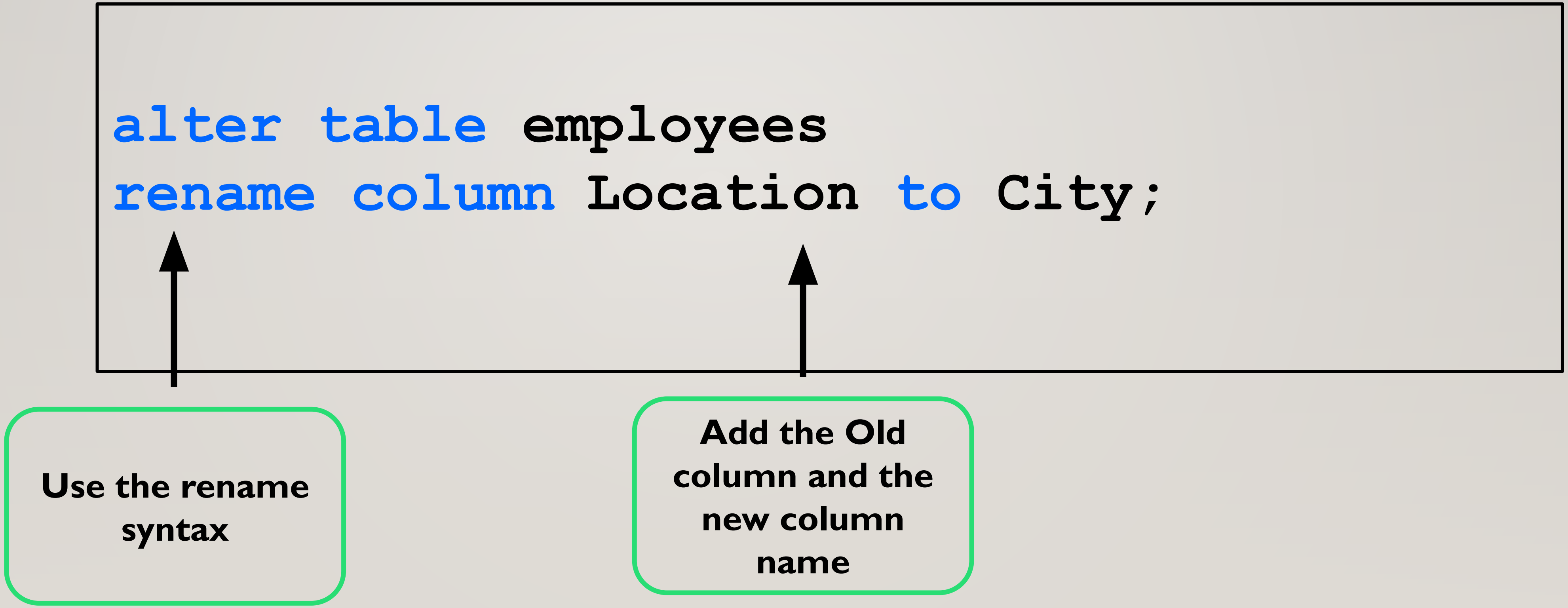


```
alter table employees
modify Location char(15);
```

Use the Modification syntax

Add the column name and Data Type

SYNTAX :ALTERING TABLE

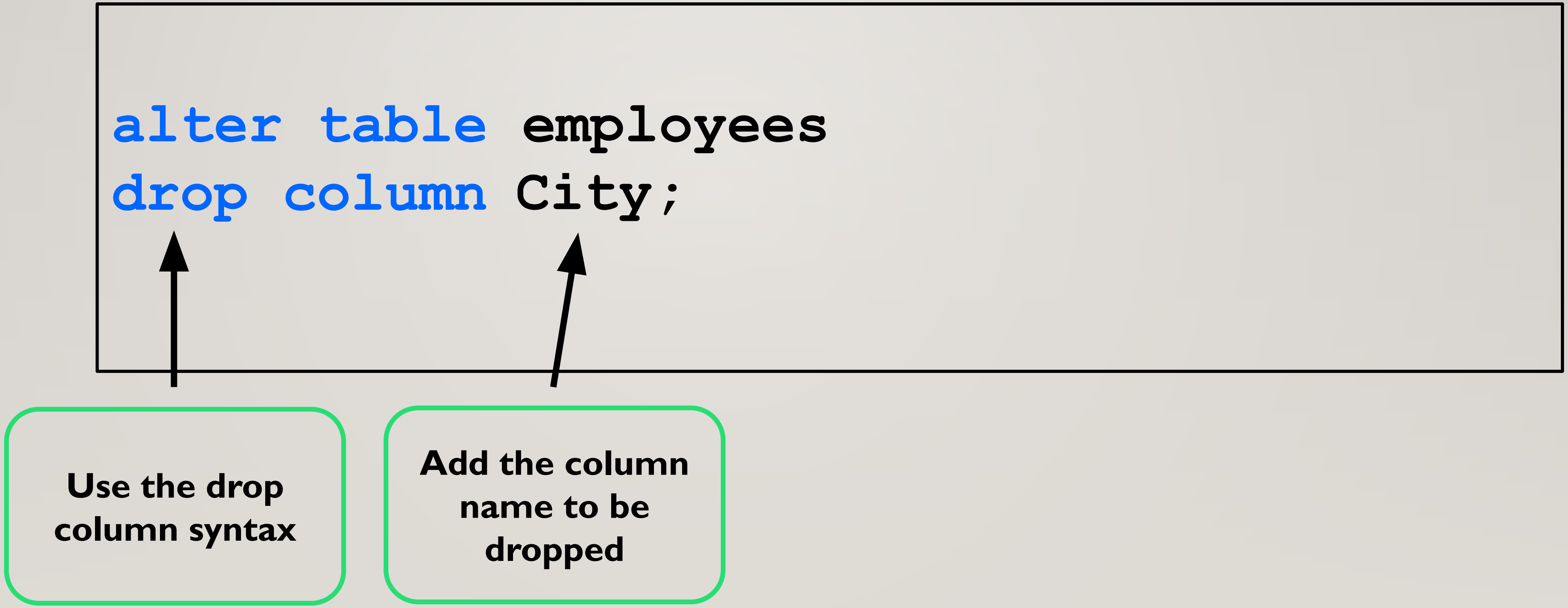


```
alter table employees
rename column Location to City;
```

Use the rename syntax

Add the Old column and the new column name

SYNTAX : ALTERING TABLE

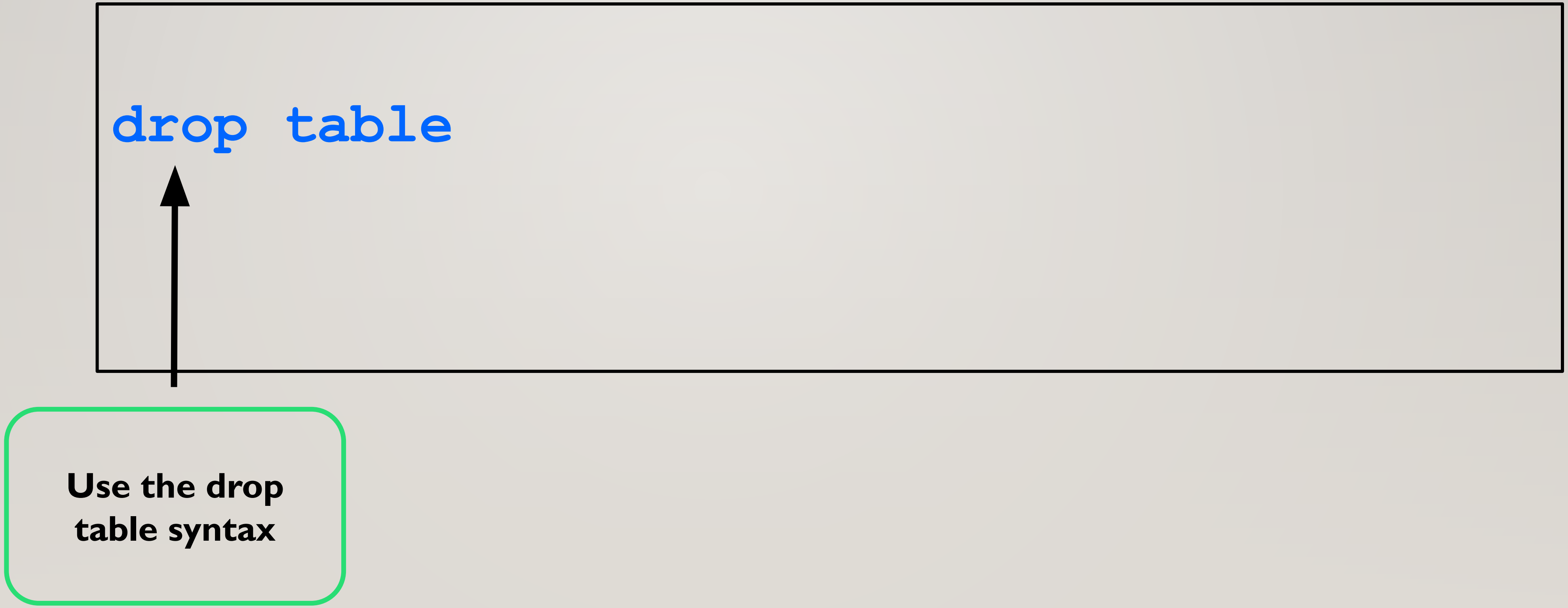


```
alter table employees
drop column City;
```

Use the drop column syntax

Add the column name to be dropped

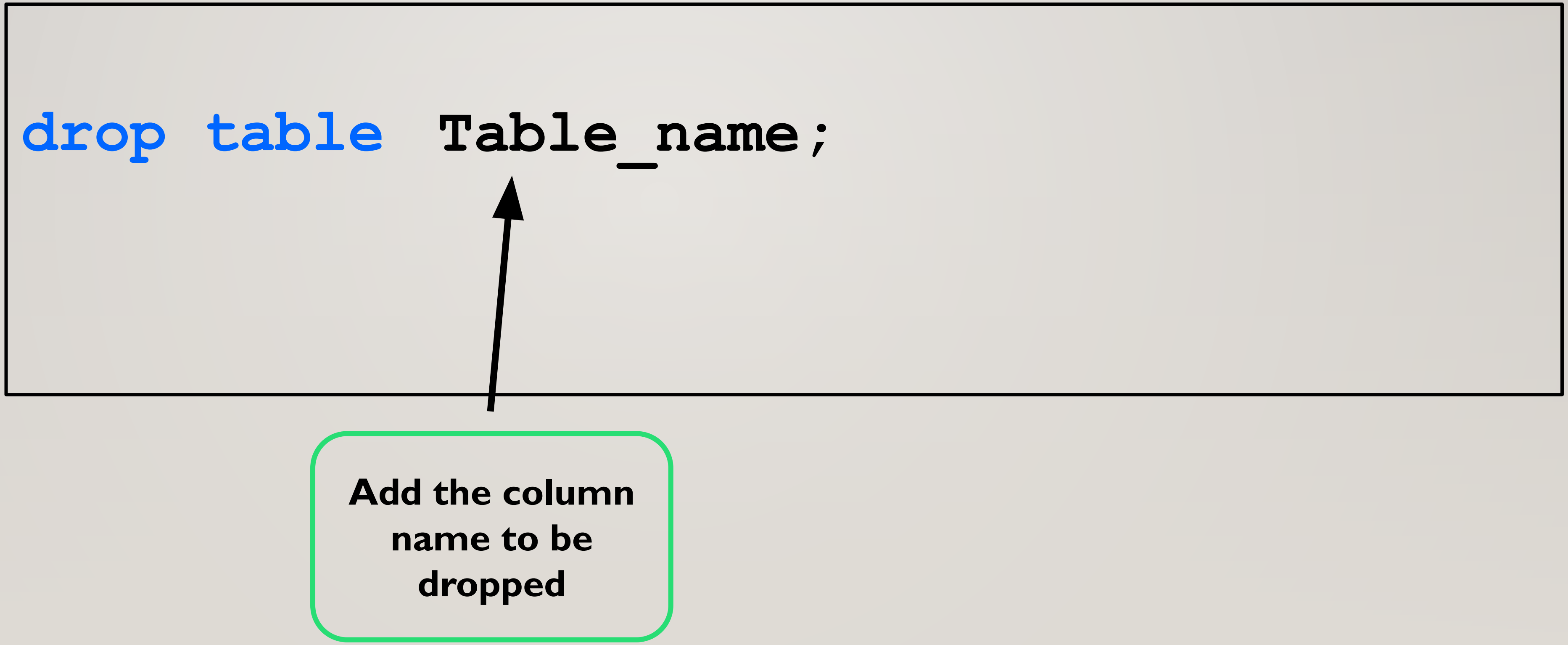
SYNTAX : DROPPING TABLE



```
drop table
```

Use the drop table syntax

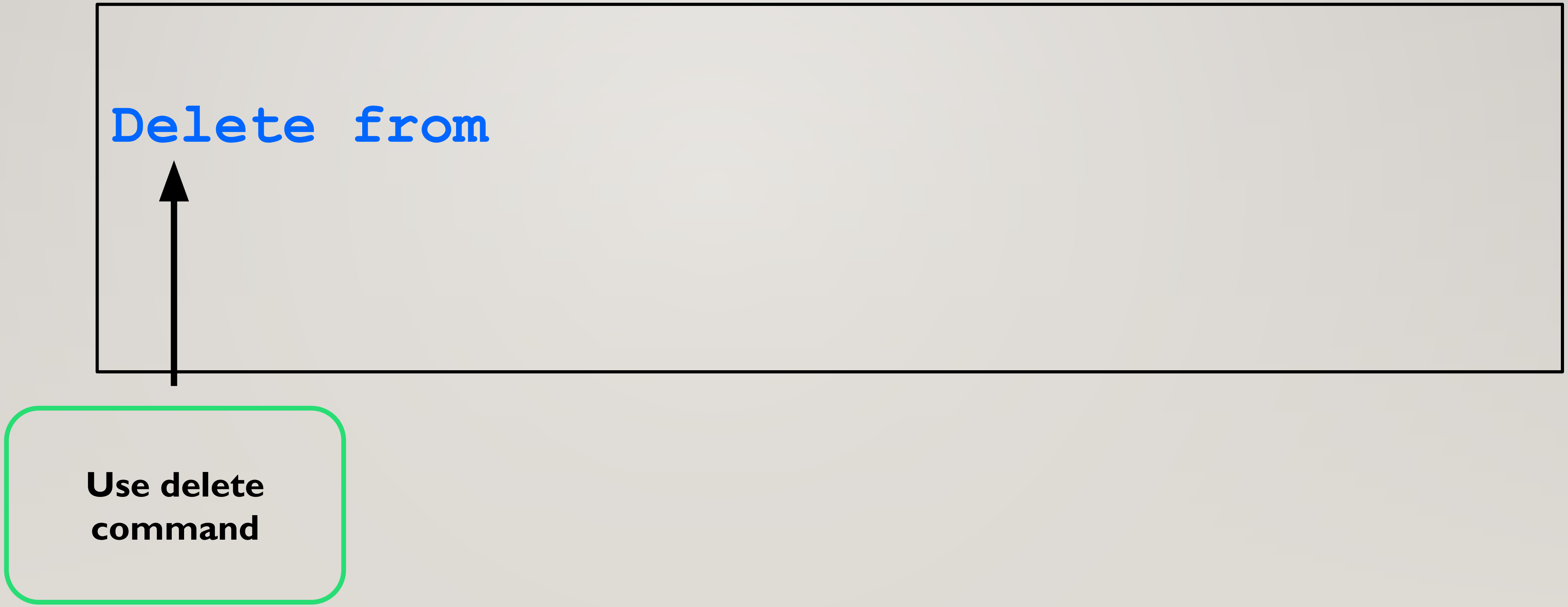
SYNTAX : DROPPING TABLE



```
drop table  Table_name;
```

Add the column name to be dropped

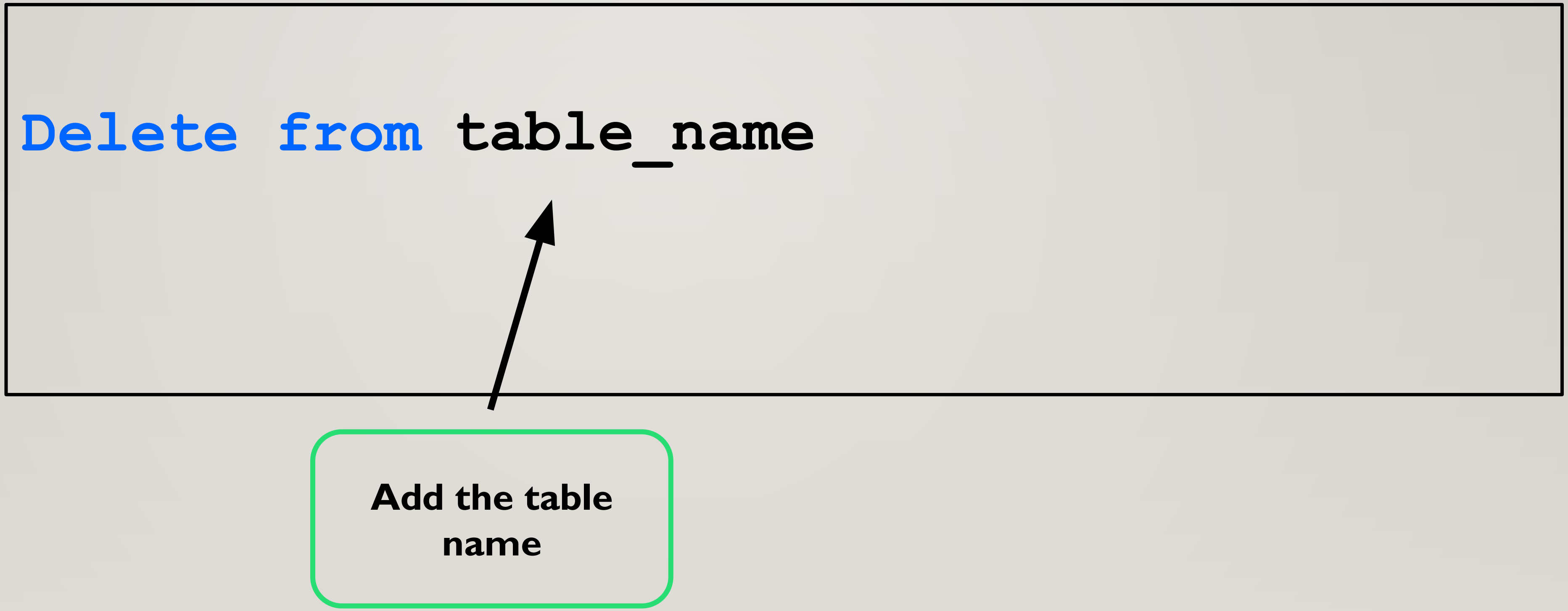
SYNTAX : DELETING RECORDS



```
Delete from
```

Use delete command

SYNTAX : DELETING RECORDS



```
Delete from table_name
```

Add the table name

SYNTAX : DELETING RECORDS

```
Delete from table_name
where
```

Use the where clause

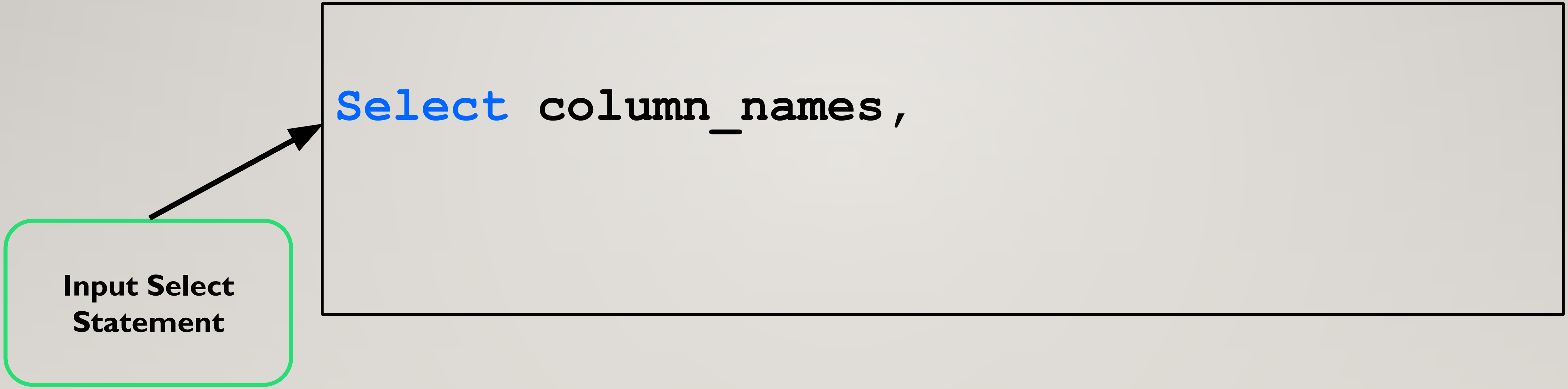
SYNTAX : DELETING RECORDS

```
Delete from table_name
where Condition;
```

Add the Condition

Handling Missing Values

SYNTAX : IFNULL



```
Select column_names,
```

Input Select Statement

SYNTAX : IFNULL

```
Select column_names,Ifnull (Column_name, value)
```

Input the If null Condition

SYNTAX : IFNULL

```
Select column_names,Ifnull (Column_name, value)
From table_name;
```

Input the Table Name

SYNTAX : IFNULL

```
Select column_names,coalesce (Column_name, value)
From table_name;
```

Input the Table Name

Date And Time Function

SYNTAX : DATE AND TIME FUNCTION

```
Select column_names,Year (Date_column)
From table_name;
```

Displays only the year of the date

SYNTAX : DATE AND TIME FUNCTION

```
Select column_names ,Month (Date_column)
From table_name;
```

Displays only the Month of the date

SYNTAX : DATE AND TIME FUNCTION

```
Select column_names, Day (Date_column)
From table_name;
```

Displays only the Day of the date

MORE FUNCTIONS

```
Minute
Hour
Second
Date add – Date add(Date, interval value unit)
Date Sub – Date sub(Date, interval value unit)
Date Diff - Date diff(end_date, start_date)
```

String Functions In SQL

SYNTAX : UPPER AND LOWER CASE

```
Select Upper(column_name) From table_name;
```

Use the Upper Function

SYNTAX : UPPER AND LOWER CASE

```sql
Select lower(column_name) From table_name;
```

Use the Lower Function

SYNTAX : LENGTH FUNCTION

```
Select length(column_name) From table_name;
```

Use the Length Function

SYNTAX : INSTRING FUNCTION

Use the instr Function

```
Select instr(column_name,’string’)
From table_name;
```

Returns the position of a string in the text

SYNTAX : SUBSTRING FUNCTION

Provide the start position of the text

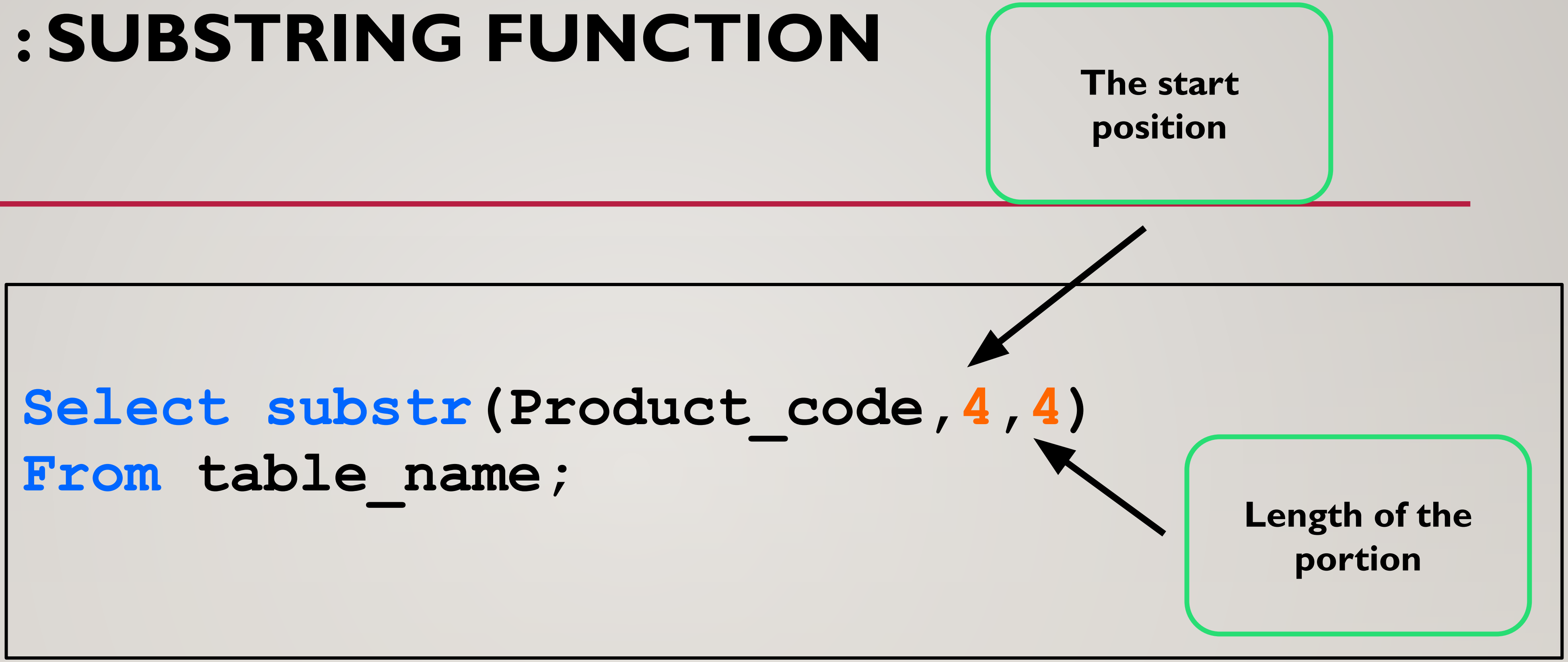
Use the substr Function

```
Select substr(column_name,Start_position,
string_length) From table_name;
```

Provide the length of the portion

Returns a portion of the input from the entire input

SYNTAX : SUBSTRING FUNCTION

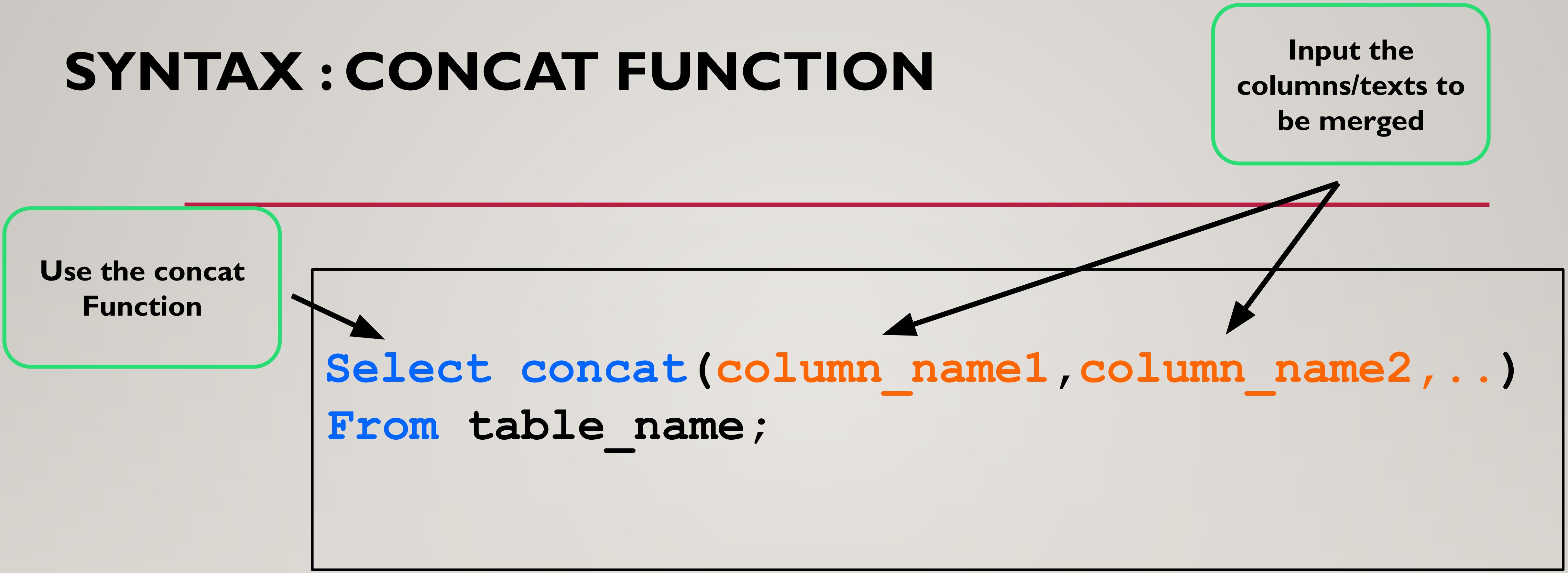


```
Select substr(Product_code,4,4)
From table_name;
```

The start position

Length of the portion

SYNTAX : CONCAT FUNCTION



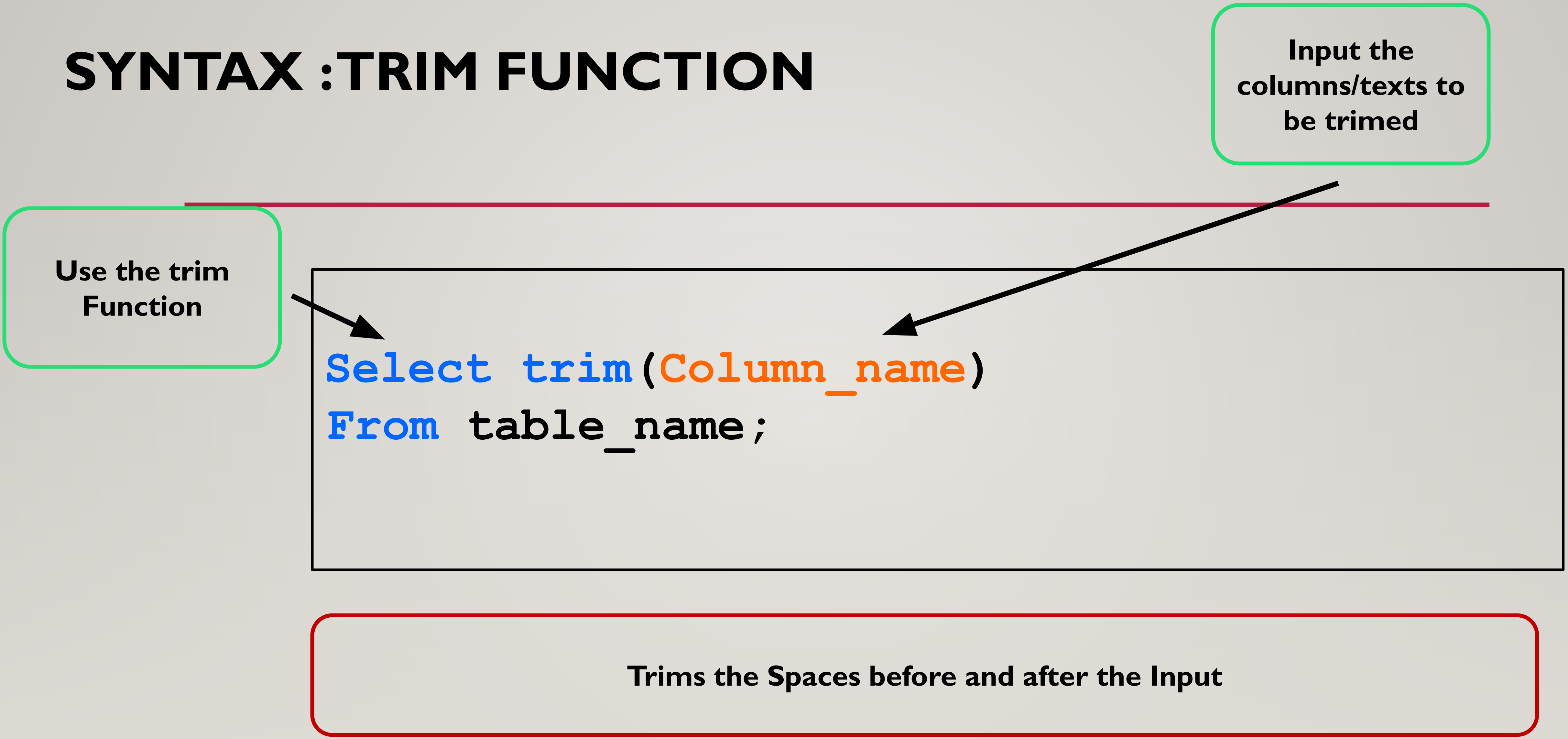
Input the columns/texts to be merged

Use the concat Function

```
Select concat(column_name1,column_name2,..)
From table_name;
```

Merges the inputs provided

SYNTAX :TRIM FUNCTION



Use the trim Function

Input the columns/texts to be trimed

```
Select trim(Column_name)
From table_name;
```

Trims the Spaces before and after the Input