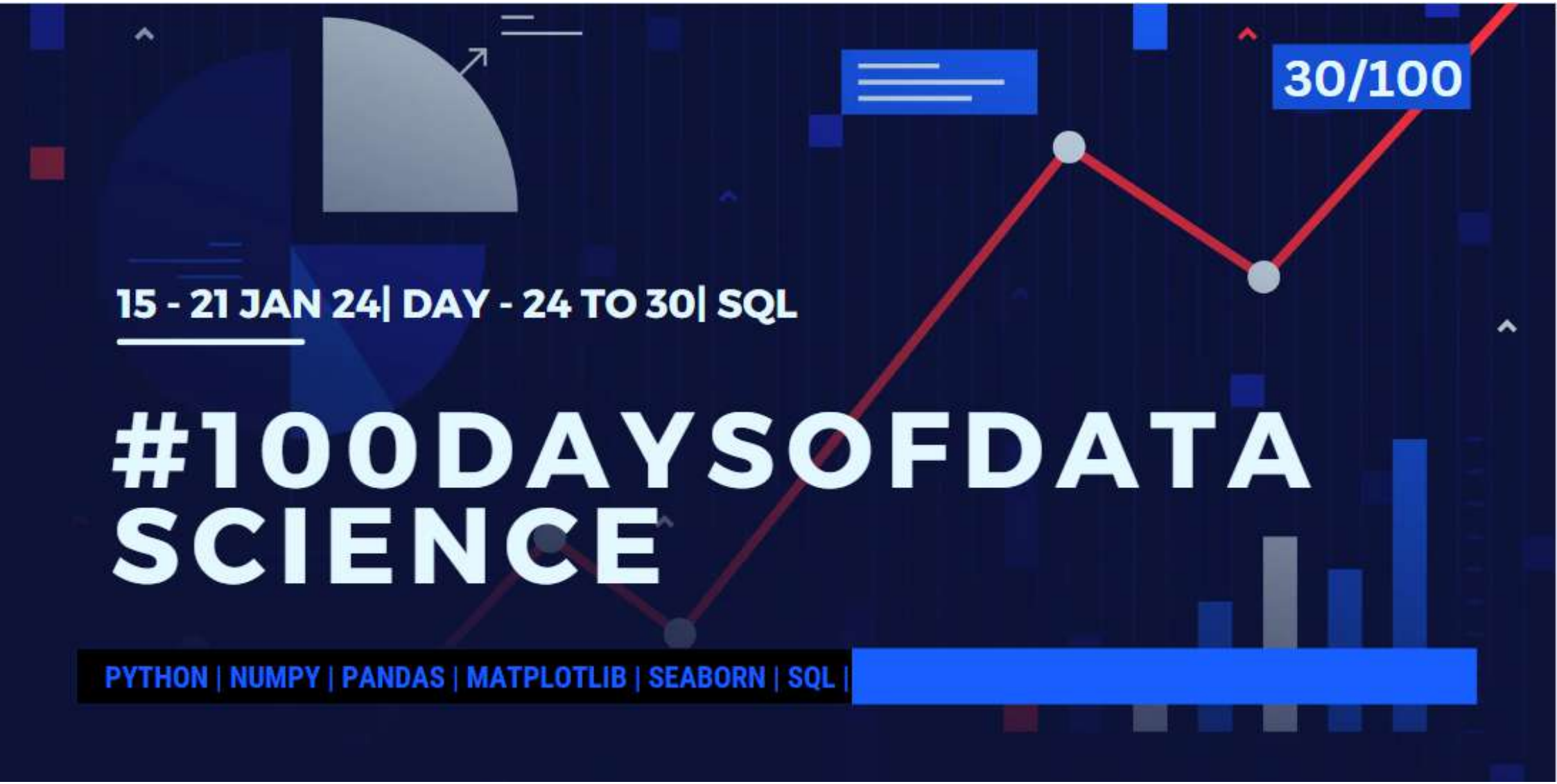

30/100
15 - 21 JAN 24| DAY - 24 TO 30| SQL
#100DAYSOFDATA SCIENCE
PYTHON | NUMPY | PANDAS | MATPLOTLIB | SEABORN | SQL |

INTRODUCTION TO SQL

- SQL is case insensitive. But it is a recommended practice to use keywords (like SELECT, UPDATE, CREATE, etc.) in capital letters
- SQL stands for Structured Query Language
- SQL is a standard language for storing, manipulating and retrieving data in databases
- SQL is an ANSI (American National Standards Institute) standard

---

SQL Data Types

- Each column in a database table is required to have a name and a data type.
- SQL developers have to decide what types of data will be stored inside each and every table column when creating a SQL table.
- The data type is a label and a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.
- For every database, data types are primarily classified into three categories.
    1. Numeric Datatypes
    2. Date and Time Database
    3. String Database

1. Numeric Data Types

- There are eleven subtypes which are given below in the table. The table contains the range of data in a particular type.

Sr. No.	Data Type	From	To
1	BigInt	-2^{63} (-9,223,372,036,854,775,808)	2^{63} -1 (9,223,372,036,854,775,807)
2	Int	-2^{31} (-2,147,483,648)	2^{31}-1 (2,147,483,647)
3	smallint	-2^{15} (-32,768)	2^{15}-1 (32,767)
4	tinyint	0	2^{8}-1(255)
5	bit	0	1
6	decimal	-10^{38}+1	10^{38}-1
7	numeric	-10^{38}+1	10^{38}-1
8	money	-922,337,203,685,477.5808	922,337,203,685,477.5808
9	smallmoney	-214,748.3647	214,748.3647
10	Float	-1.79E+308	1.79E+308
11	Real	-3.40E+38	3.40E+38

2. String Data Types

- The subtypes are given in below table –

Sr. No.	Data Type	Description
1	char	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
2	varchar	It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
3	Text	It holds a string that can contain a maximum length of 255 characters.

3. Date and Time Data Type

- The details are given in the below table.

Sr. No.	Data Type	Description
1	DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
2	DATETIME	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
3	TIMESTAMP	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
4	TIME	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
5	YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

SQL SELECT Statement

- It is used to access the records from one or more database tables and views.
- It also retrieves the selected data that follow the conditions we want.
- The data returned is stored in a result table, called the result-set.
- **Syntax:**

  Here, column1, column2, ... are the field names of the table you want to select data from.

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

- **Example:**

  The following SQL statement selects all the columns from the "products" table:

```
8 •  select * from products;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	
S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front su...	7933	4
S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels; steering function; detail...	7305	9
S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi logos and insignias, saddle b...	6625	6
S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles	1:10	Red Start Diecast	Model features, official Harley Davidson logos a...	5582	9
S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Turnable front wheels; steeri...	3252	8
S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Turnable front wheels; steeri...	6791	1
S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	Hood, doors and trunk all open to reveal highly ...	68	9
S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels; steering function; detail...	3619	9
S12_1666	1958 Setra Bus	Trucks and Buses	1:12	Welly Diecast Productions	Model features 30 windows, skylights & glare re...	1579	7
S12_2823	2002 Suzuki XREO	Motorcycles	1:12	Unimax Art Galleries	Official logos and insignias, saddle bags located ...	9997	6
S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions	1:18 scale die-cast about 10" long doors open, ...	6906	8
S12_3380	1968 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	1:12 scale model of a 1968 Dodge Charger. Ho...	9123	7
S12_3891	1969 Ford Falcon	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels; steering function; detail...	1049	8
S12_3990	1970 Plymouth Hemi Cuda	Classic Cars	1:12	Studio M Art Models	Very detailed 1970 Plymouth Cuda model in 1:1...	5663	3

- **Columns Example:**

The following SQL statement selects the "productName", “buyPrice” and "productLine" columns from the "products" table:

```
7 •  select productName,buyPrice,productLine from products;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

productName	buyPrice	productLine
1969 Harley Davidson Ultimate Chopper	48.81	Motorcycles
1952 Alpine Renault 1300	98.58	Classic Cars
1996 Moto Guzzi 1100i	68.99	Motorcycles
2003 Harley-Davidson Eagle Drag Bike	91.02	Motorcycles
1972 Alfa Romeo GTA	85.68	Classic Cars
1962 LanciaA Delta 16V	103.42	Classic Cars
1968 Ford Mustang	95.34	Classic Cars
2001 Ferrari Enzo	95.59	Classic Cars
1958 Setra Bus	77.90	Trucks and Buses
2002 Suzuki XREO	66.27	Motorcycles
1969 Corvair Monza	89.14	Classic Cars
1968 Dodge Charger	75.16	Classic Cars
1969 Ford Falcon	83.05	Classic Cars

SQL WHERE Clause

- A WHERE clause in SQL is a data manipulation language statement.
- It is used to fetch data according to particular criteria.
- WHERE clauses are not mandatory clauses of SQL DML statements. But it can be used to limit the number of rows affected by a SQL DML statement or returned by a query.
- WHERE clause is used in SELECT, UPDATE, DELETE statement etc.
- It acts as a gatekeeper, ensuring only rows that meet specific conditions are included in the output.
- It enables you to create targeted queries that return only the relevant results.

- **Syntax:**

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

- **Example:**

  The following SQL statement selects all the products from the productLine "Classic Cars", in the "Products" table:

```
7 •  select * from products where productLine='Classic Cars';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

productCode	productName	productLine	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP
S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels; steering function; detail...	7305	98.58	214.30
S10_4757	1972 Alfa Romeo GTA	Classic Cars	1:10	Motor City Art Classics	Features include: Turnable front wheels; steeri...	3252	85.68	136.00
S10_4962	1962 LanciaA Delta 16V	Classic Cars	1:10	Second Gear Diecast	Features include: Turnable front wheels; steeri...	6791	103.42	147.74
S12_1099	1968 Ford Mustang	Classic Cars	1:12	Autoart Studio Design	Hood, doors and trunk all open to reveal highly ...	68	95.34	194.57
S12_1108	2001 Ferrari Enzo	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels; steering function; detail...	3619	95.59	207.80
S12_3148	1969 Corvair Monza	Classic Cars	1:18	Welly Diecast Productions	1:18 scale die-cast about 10" long doors open, ...	6906	89.14	151.08
S12_3380	1968 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	1:12 scale model of a 1968 Dodge Charger. Ho...	9123	75.16	117.44
S12_3891	1969 Ford Falcon	Classic Cars	1:12	Second Gear Diecast	Turnable front wheels; steering function; detail...	1049	83.05	173.02
S12_3990	1970 Plymouth Hemi Cuda	Classic Cars	1:12	Studio M Art Models	Very detailed 1970 Plymouth Cuda model in 1:1...	5663	31.92	79.80
S12_4675	1969 Dodge Charger	Classic Cars	1:12	Welly Diecast Productions	Detailed model of the 1969 Dodge Charger. Thi...	7323	58.73	115.16
S18_1129	1993 Mazda RX-7	Classic Cars	1:18	Highway 66 Mini Classics	This model features, opening hood, opening do...	3975	83.51	141.54
S18_1589	1965 Aston Martin DB5	Classic Cars	1:18	Classic Metal Creations	Die-cast model of the silver 1965 Aston Martin ...	9042	65.96	124.44
S18_1889	1948 Porsche 356-A Roa...	Classic Cars	1:18	Gearbox Collectibles	This precision die-cast replica features opening ...	8826	53.90	77.00

- **Operators in The WHERE Clause**

  You can use other operators than the = operator to filter the search.
  The following operators can be used in the WHERE clause:

Sr. No.	Operator	Description
1	=	Equal
2	>	Greater than
3	<	Less than
4	>=	Greater than or equal
5	<=	Less than or equal
6	<> !=	Not equal. Note: In some versions of SQL this operator may be written as !=
7	BETWEEN	Between a certain range
8	LIKE	Search for a pattern
9	IN	To specify multiple possible values for a column

1. Equal:

The following SQL statement selects all the employees from the OfficeCode = "1", in the "Employees" table:

```
10 •  select * from employees where officeCode=1;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep

---

2. Greater than:

The following SQL statement selects all the employees from the OfficeCode > "3", in the "Employees" table:

```
10 •  select * from employees where officeCode>3;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep

3. Less than:

The following SQL statement selects all the employees from the OfficeCode < "3", in the "Employees" table:

```
10 •   select * from employees where officeCode<3;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep

---

4. Greater than or equal:

The following SQL statement selects all the employees from the OfficeCode >= "5", in the "Employees" table:

```
10 •   select * from employees where officeCode>=5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep
1612	Marsh	Peter	x102	pmarsh@classicmodelcars.com	6	1088	Sales Rep
1619	King	Tom	x103	tking@classicmodelcars.com	6	1088	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep
1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	Sales Rep

---

5. Less than or Equal:

The following SQL statement selects all the employees from the OfficeCode <= "3", in the "Employees" table:

```
10 •   select * from employees where officeCode<=3;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep

6. Not Equal:

The following SQL statement selects all the employees from the OfficeCode <> "1", in the "Employees" table:

```
10 •  select * from employees where officeCode<>1;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep
1612	Marsh	Peter	x102	pmarsh@classicmodelcars.com	6	1088	Sales Rep
1619	King	Tom	x103	tking@classicmodelcars.com	6	1088	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep
1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	Sales Rep
1702	Gerard	Martin	x2312	mgerard@classicmodelcars.com	4	1102	Sales Rep

---

7. BETWEEN:

a. The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
b. The BETWEEN operator is inclusive: begin and end values are included.
c. The BETWEEN Condition will return the records where the expression is within the range of value1 and value2.
   The following SQL statement selects all the employees from the OfficeCode between "3" and “6”, in the "Employees" table:

```
10 •  select * from employees where officeCode between 3 and 7;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep
1612	Marsh	Peter	x102	pmarsh@classicmodelcars.com	6	1088	Sales Rep
1619	King	Tom	x103	tking@classicmodelcars.com	6	1088	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep
1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	Sales Rep
1702	Gerard	Martin	x2312	mgerard@classicmodelcars.com	4	1102	Sales Rep

8. IN:

a. The IN operator allows you to specify multiple values in a WHERE clause.
b. The IN operator is a shorthand for multiple OR conditions.
c. A sub-query or list of values must be specified in the parenthesis
d. We can use the IN operator with the INSERT, SELECT, UPDATE, and DELETE queries in the SQL database.
   The following SQL statement selects all the employees from the OfficeCode in "3" or “5” or “7”, in the "Employees" table:

```
10 •   select * from employees where officeCode in (3,5,7);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep
1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	Sales Rep

---

9. LIKE:

a. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
b. SQL wildcards are special characters used in SQL queries to match patterns in the data.
c. There are two wildcards often used in conjunction with the LIKE operator:
   - The percent sign % represents zero, one, or multiple characters
   - The underscore sign _ represents one, single character

   The following SQL statement selects all the employees where firstName starts with “a”, in the "Employees" table:

```
11 •   select * from employees where firstName like "a%";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep

The following SQL statement selects all the employees where firstName ends with “y”, in the "Employees" table:

```
11 •   select * from employees where firstName like "%y";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep

SQL AND, OR and NOT Operator

- The WHERE clause can be combined with AND, OR, and NOT operators.

1. AND Operator:

a. The WHERE clause can contain one or many AND operators.
b. The AND operator is used to filter records based on more than one condition
c. **Syntax:**

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

d. **Example:**

The following SQL statement selects all the employees where firstName ends with “y” and officeCode = “7”, in the "Employees" table:

```
12 •    select * from employees where officeCode = "7" and firstName like "%y";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep

2. OR Operator:

a. The WHERE clause can contain one or more OR operators.
b. The OR operator is used to filter records based on more than one condition
c. **Syntax:**

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

d. **Example:**

The following SQL statement selects all the employees where firstName ends with “y” or officeCode = “7”, in the "Employees" table:

```
12 •  select * from employees where officeCode = "7" or firstName like "%y";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep

---

AND vs OR vs NOT

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

---

3. NOT Operator:
   a. The NOT operator is used in combination with other operators to give the opposite result, also called the negative result.
   b. Alternatively you can use <> (Not Operator) to get the same result
   c. **Syntax:**

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

d. **Example:**

The following SQL statement selects all the employees where officeCode not equal to “7”, in the "Employees" table:

```
12 •  select * from employees where not officeCode = "7";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep

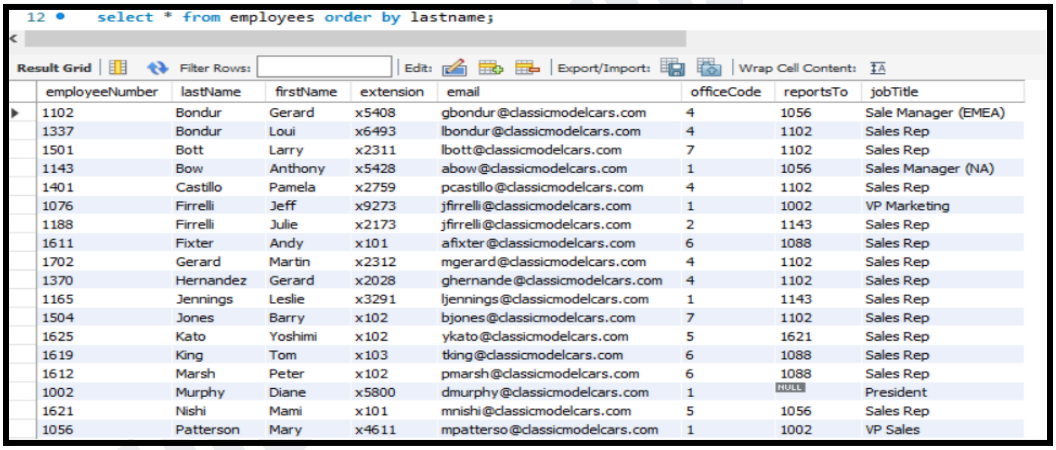
SQL ORDER BY Keyword

- 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.
- **Syntax:**

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

- **Example:**

The following SQL statement selects all the employees sorting lastname ascending, in the "Employees" table:

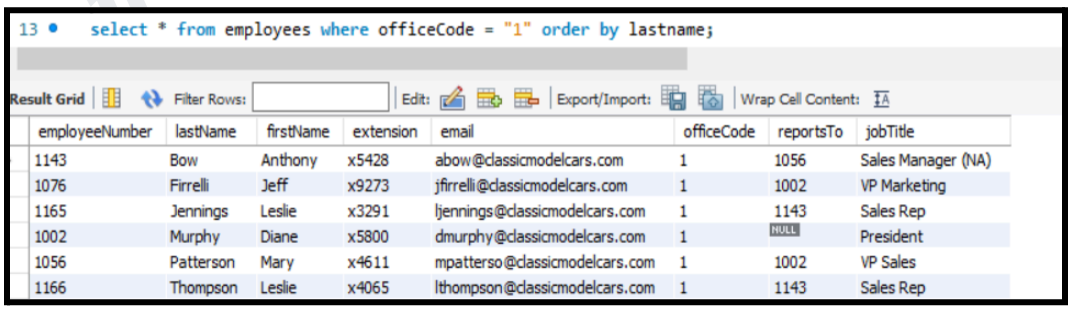


```
12 •  select * from employees order by lastname;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep
1702	Gerard	Martin	x2312	mgerard@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	Sales Rep
1619	King	Tom	x103	tking@classicmodelcars.com	6	1088	Sales Rep
1612	Marsh	Peter	x102	pmarsh@classicmodelcars.com	6	1088	Sales Rep
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales

The following SQL statement selects all the employees where officeCode = “1” and sorting lastname descending, in the "Employees" table:



```
13 •  select * from employees where officeCode = "1" order by lastname;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep

SQL SELECT DISTINCT Statement

- The SELECT DISTINCT statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
- The SELECT DISTINCT statement is used to return only distinct (different) values.
- **Syntax:**

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

- **Example:**

The following SQL statement selects Unique cities, in the "Customers" table:

```
11 •   select distinct city, country from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city	country
Nantes	France
Las Vegas	USA
Melbourne	Australia
Stavern	Norway
San Rafael	USA
Warszawa	Poland
Frankfurt	Germany
San Francisco	USA
NYC	USA
Madrid	Spain
Luleå	Sweden
Kobenhavn	Denmark
Lyon	France
Singapore	Singapore
Allentown	USA

The following SQL statement selects count of unique cities, in the "Customers" table:

```
11 •   select count(distinct city) from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(distinct city)
96

SQL LIMIT Clause

- The LIMIT in SQL is a clause that enables multi-page outcomes or SQL pagination to be easily coded and is very helpful on large tables.
- The LIMIT clause is used to specify the number of records to return.
- The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.
- The LIMIT keyword is used to LIMIT the number of rows of a result set returned
- Any number from zero (0) and up can be the LIMIT number. No rows are returned from the set result if zero (0) is set as the LIMIT

- **Syntax:**

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

- **Example:**

The following SQL statement select all columns till 12 records by creditLimit Ascending, in the "Customers" table:

```
15 •  select * from customers order by creditLimit desc limit 12;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	addressLine2	city	state	postalCode	country
141	Euro+ Shopping Channel	Freyre	Diego	(91) 555 94 44	C/ Moralzarzal, 86	NULL	Madrid	NULL	28034	Spain
124	Mini Gifts Distributors Ltd.	Nelson	Susan	4155551450	5677 Strong St.	NULL	San Rafael	CA	97562	USA
298	Vida Sport, Ltd	Holz	Mihael	0897-034555	Grenzacherweg 237	NULL	Genève	NULL	1203	Switzerla
151	Muscle Machine Inc	Young	Jeff	2125557413	4092 Furth Circle	Suite 400	NYC	NY	10022	USA
187	AV Stores, Co.	Ashworth	Rachel	(171) 555-1555	Fauntleroy Circus	NULL	Manchester	NULL	EC2 5NT	UK
146	Saveley & Henriot, Co.	Saveley	Mary	78.32.5555	2, rue du Commerce	NULL	Lyon	NULL	69004	France
286	Marta's Replicas Co.	Hernandez	Marta	6175558555	39323 Spinnaker Dr.	NULL	Cambridge	MA	51247	USA
386	L'ordine Souveniers	Moroni	Maurizio	0522-556555	Strada Provinciale 124	NULL	Reggio Emilia	NULL	42100	Italy
227	Heintze Collectables	Ibsen	Palle	86 21 3555	Smagsloget 45	NULL	Århus	NULL	8200	Denmark
259	Toms Spezialitäten, Ltd	Pfalzheim	Henriette	0221-5554327	Mehrheimerstr. 369	NULL	Köln	NULL	50739	Germany
278	Rovelli Gifts	Rovelli	Giovanni	035-640555	Via Ludovico il Moro 22	NULL	Bergamo	NULL	24100	Italy
119	La Rochelle Gifts	Labrune	Janine	40.67.8555	67, rue des Cinquant...	NULL	Nantes	NULL	44000	France

SQL MAX() & MIN() Functions

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

- **Syntax:**
  1. MIN()

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

2. MAX()

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

- **Example:**
  1. MIN()

```
15 •    select min(creditLimit)from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

min(creditLimit)
0.00

2. MAX()

```
16 •    select max(creditLimit)from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

max(creditLimit)
227600.00

SQL SUM() & AVG() FUNCTIONS

- The SUM() function returns the total sum of a numeric column.
- The AVG() function returns the average value of a numeric column.

- **Syntax:**
  1. SUM()

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

2. AVG()

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

- **Example:**
  1. SUM()

```
16 •    select sum(creditLimit)from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum(creditLimit)
8254400.00

2. AVG()

```
16 •    select avg(creditLimit)from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

avg(creditLimit)
67659.016393

SQL COUNT() & LEN() FUNCTIONS

- The COUNT() function returns the number of rows that matches a specified criteria.
- The LEN() function returns the length of the value in a text field.

- **Syntax:**
  1. COUNT()

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

2. LEN()

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

- **Example:**
  1. COUNT()

The following SQL statement count number of creditLimit, in the "Customers" table:

```
16 •    select count(creditLimit)from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

count(creditLimit)
122

2. LEN()

The following SQL statement select number of char present in the address, in the "Customers" table:

```
16 •    select city, customerName, creditLimit, length(addressLine1) from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city	customerName	creditLimit	length(addressLine1)
Nantes	Atelier graphique	21000.00	14
Las Vegas	Signal Gift Stores	71800.00	15
Melbourne	Australian Collectors, Co.	117300.00	17
Nantes	La Rochelle Gifts	118200.00	28
Stavern	Baane Mini Imports	81700.00	22
San Rafael	Mini Gifts Distributors Ltd.	210500.00	15
Warszawa	Havel & Zbyszek Co	0.00	15
Frankfurt	Blauer See Auto, Co.	59700.00	13
San Francisco	Mini Wheels Co.	64600.00	25
NYC	Land of Toys Inc.	114900.00	23

SQL ALIAS Clause

- SQL aliases are used to give a table, or a column in a table, a temporary name
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of the query.
- The AS keyword is used to give columns or tables a temporary name that can be used to identify that column or table later.
- An SQL alias is useful for simplifying your queries and making the query and its result more readable.

- **Syntax:**

```
SELECT column_name AS alias_name
FROM table_name;
```

- **Example:**

The following SQL statement renaming city to City, customerName to Name, creditLimit to Limit, in the "Customers" table:

```
16 •   select city as City, customerName as "Name", creditLimit as "Limit" from customers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

City	Name	Limit
Nantes	Atelier graphique	21000.00
Las Vegas	Signal Gift Stores	71800.00
Melbourne	Australian Collectors, Co.	117300.00
Nantes	La Rochelle Gifts	118200.00
Stavern	Baane Mini Imports	81700.00
San Rafael	Mini Gifts Distributors Ltd.	210500.00
Warszawa	Havel & Zbyszek Co	0.00
Frankfurt	Blauer See Auto, Co.	59700.00
San Francisco	Mini Wheels Co.	64600.00
NYC	Land of Toys Inc.	114900.00
Madrid	Euro+ Shopping Channel	227600.00
Luleå	Volvo Model Replicas, Co	53100.00
Kobenhavn	Danish Wholesale Imports	83400.00
Lyon	Saveley & Henriot, Co.	123900.00
Singapore	Dragon Souveniers, Ltd.	103800.00

SQL GROUP BY Statement

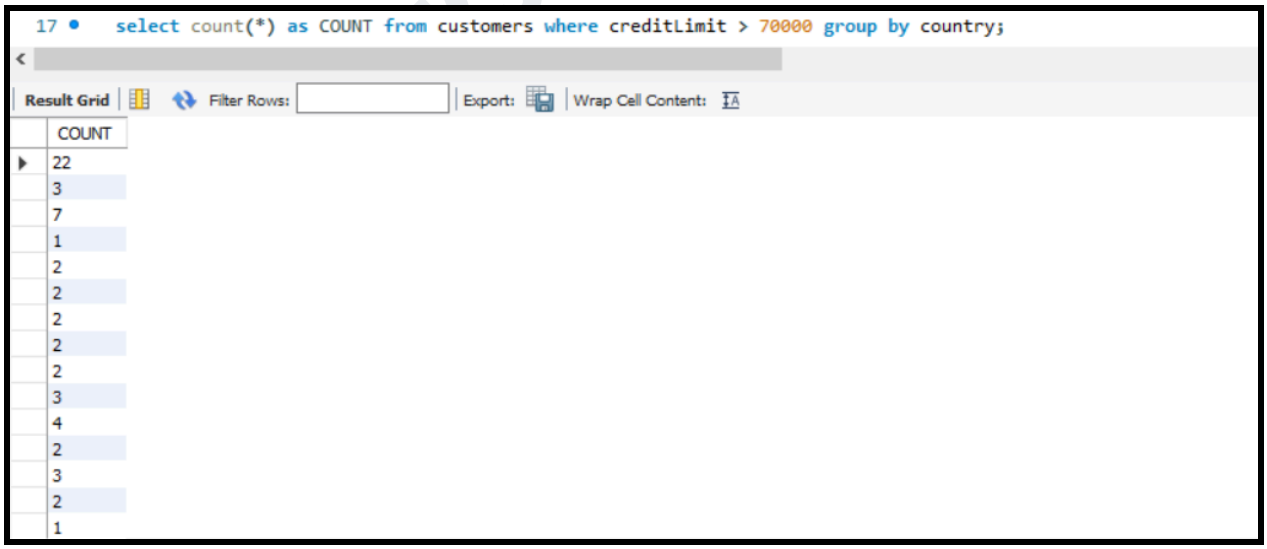
- GROUP BY is a SQL query clause used to group data rows based on one or more columns in a table.
- The query returns aggregated data instead of individual rows when using GROUP BY.
- The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

- **Syntax:**

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

- **Example:**

The following SQL statement selects count of customers where creditLimit > 70000 and grouping them according to their country, in the "Customers" table:



SQL HAVING Clause

- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions
- The SQL HAVING clause is similar to the WHERE clause; both are used to filter rows in a table based on specified criteria.
- The HAVING keyword was introduced because the WHERE clause fails when used with aggregate functions.
- With the HAVING clause, you can arrange the data in your database into many groups when you use it with the GROUP BY keyword

- **Syntax:**

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

- **Example:**

```
14 •  select count(*) as total, productLine from products group by productLine having count(*) > 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

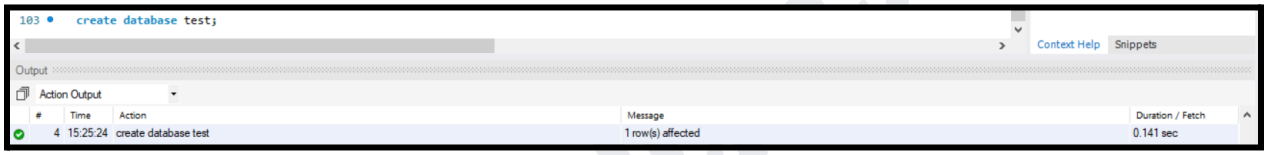
total	productLine
38	Classic Cars
13	Motorcycles
12	Planes
9	Ships
11	Trucks and Buses
24	Vintage Cars

SQL CREATE DATABASE Statement

- The CREATE DATABASE statement is used to create a new SQL database
- **Syntax:**

```
CREATE DATABASE databasename;
```

- **Example:**

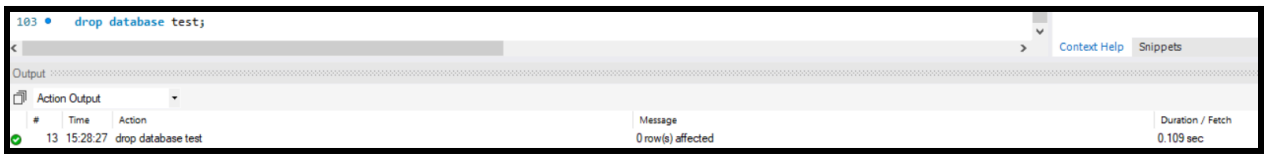


SQL DROP DATABASE Statement

- The DROP DATABASE statement is used to drop an existing SQL database.
- **Syntax:**

```
DROP DATABASE databasename;
```

- **Example:**



SQL CREATE TABLE Statement

- The CREATE TABLE statement is used to create a new table in a database
- The column parameters specify the names of the columns of the table.
- The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

- **Syntax:**

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

- **Example:**

The following SQL statement create table task where there is task_id, title, start_date, due_date, status, priority, description, created_at, in the "test" database:

```
37  create table task(
38      task_id int auto_increment primary key,
39      title varchar(255) not null,
40      start_date date,
41      due_date date,
42      status tinyint not null,
43      priority tinyint not null,
44      description text,
45      created_at timestamp default current_timestamp
46  ) engine = InnoDB;
47  select * from task;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

task_id	title	start_date	due_date	status	priority	description	created_at
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL DROP TABLE Statement

- The DROP TABLE statement is used to drop an existing table in a database.
- **Syntax:**

```
DROP TABLE table_name;
```

- **Example:**

```
49 •   drop table if exists task;
50
```

SQL ALTER TABLE Statement

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.
- **Syntax:**

1. ADD Columns

```
ALTER TABLE table_name
ADD column_name datatype;
```

2. DROP Columns

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

3. ALTER/MODIFY Columns

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

SQL INSERT INTO Statement

- The INSERT query serves the fundamental purpose of introducing new records (rows) into an existing database table.
- It's the essential tool for populating tables with fresh information, ensuring they stay up-to-date and reflect current data needs.

- **Syntax:**

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

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

- **Example:**

The following SQL statement insert value into table customers where there is id, name, age, address, salary in the "test":

```
INSERT INTO CUSTOMERS VALUES
(1, 'Ramesh', 32, 'Ahmedabad', 2000.00),
(2, 'Khilan', 25, 'Delhi', 1500.00),
(3, 'Kaushik', 23, 'Kota', 2000.00),
(4, 'Chaitali', 25, 'Mumbai', 6500.00),
(5, 'Hardik', 27, 'Bhopal', 8500.00),
(6, 'Komal', 22, 'Hyderabad', 4500.00),
(7, 'Muffy', 24, 'Indore', 10000.00);
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	37	Ahmedabad	5000.00
2	Khilan	30	Delhi	4500.00
3	Kaushik	28	Kota	5000.00
4	Chaitali	30	Mumbai	9500.00
5	Hardik	32	Bhopal	11500.00
6	Komal	27	Pune	7500.00
7	Muffy	29	Indore	13000.00

SQL Update Statement

- The UPDATE query is specifically designed to alter the values of columns within records that already exist in a database table. It's a powerful tool for maintaining data accuracy and keeping information up-to-date.
- **Syntax:**

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

- **Example:**

The following SQL statement updates value of table customers where there is id, name, age, address, salary in the "test":

```
UPDATE CUSTOMERS
SET ADDRESS = 'Pune', SALARY = 1000.00
WHERE NAME = 'Ramesh';
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	37	Pune	1000.00

SQL Delete Statement

- The DELETE query is designed to permanently erase specific rows or records from a database table. It's essential for maintaining data hygiene, removing outdated information, and optimizing storage space.
- **Syntax:**

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

- **Example:**

The following SQL statement delete rows of table customers where there is id, name, age, address, salary in the "test":

```
DELETE FROM CUSTOMERS WHERE AGE > 25;
```

ID	NAME	AGE	ADDRESS	SALARY
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
7	Muffy	24	Indore	10000.00

SQL Key and Constraints

- Keys and constraints act as guardians of data integrity and consistency.
- They establish rules and restrictions that guide how data is organized and maintained within tables, ensuring its accuracy and reliability.
- It prevents data duplication, inconsistencies, and violations of business rules.
- It optimizes query performance through indexing and relationships.
- It ensures data quality and accuracy, reducing errors and promoting confidence in decision-making.

- Types of Keys:

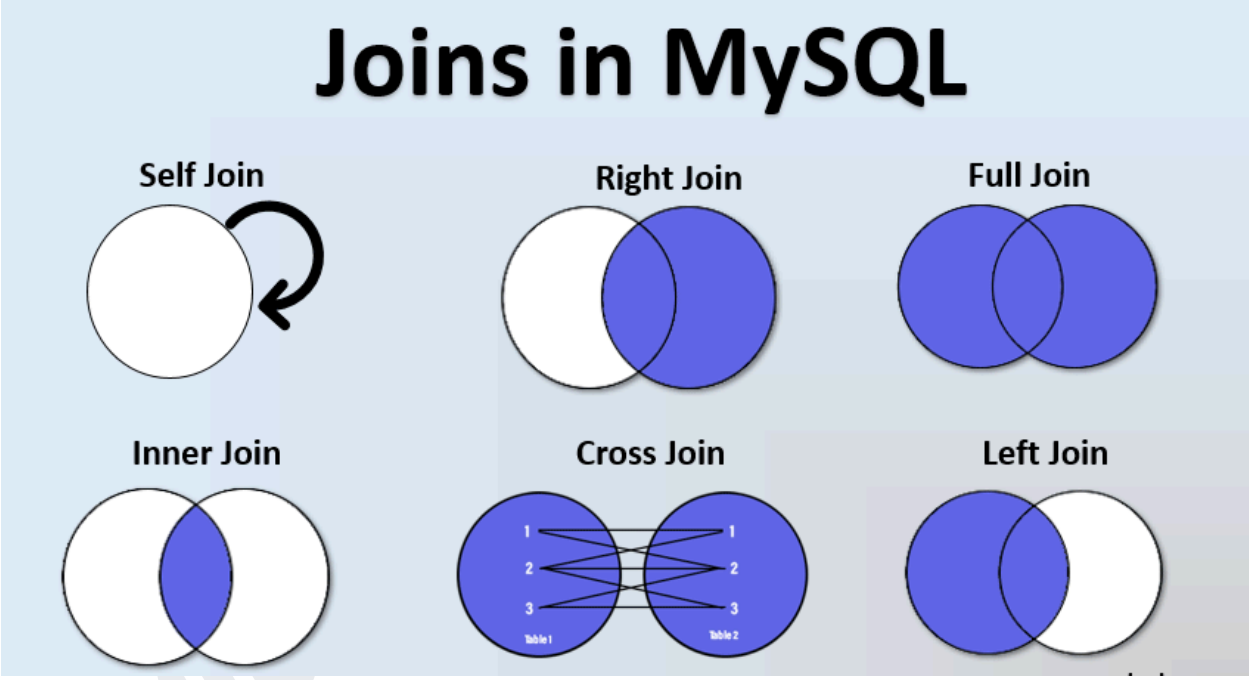
Sr. No.	Name	Definition
1	Primary Key	A unique column (or combination of columns) that unequivocally identifies each row in a table. It acts as the table's main index for accessing and joining data.
2	Foreign Key	A column (or set of columns) in one table that references the primary key of another table, establishing relationships between data and preventing inconsistencies.
3	Unique Key	A column (or set of columns) that must hold unique values within a table, ensuring data exclusivity and preventing duplicates.
4	Composite Key	Composite Key is a key that can be defined on two or more columns in a table to uniquely identify any record. It can also be described as a Primary Key created on multiple columns.

- Types of Constraints:

Sr. No.	Name	Definition
1	Not Null	Ensures a column cannot contain null values, mandating data presence for critical fields.
2	Check	Validates data based on specified conditions, guaranteeing its adherence to defined rules.
3	Default	Assigns a default value to a column if no value is provided during insertion, maintaining consistency.

SQL Joins

- In the world of databases, joins play a pivotal role in bridging isolated tables and crafting comprehensive views of interconnected data.
- They empower you to combine information from multiple tables based on shared attributes, unlocking insights that reside beyond individual tables.
- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.



- Types of Joins:

1. Inner Join
2. Self Join
3. Left Join
4. Right Join
5. Full Join
6. Cross Join

INNER JOIN

- The most common join type, it only includes rows where matched values exist in both tables.
- Ideal for retrieving related data that exists in both tables.
- **Syntax:**

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

- **Example:**

```
SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
INNER JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

ID	NAME	AMOUNT	DATE
3	Kaushik	3000.00	2009-10-08 00:00:00
3	Kaushik	1500.00	2009-10-08 00:00:00
2	Khilan	1560.00	2009-11-20 00:00:00
4	Chaitali	2060.00	2008-05-20 00:00:00

SELF JOIN

- A unique join that joins a table to itself, enabling comparisons or relationships within a single table.
- **Syntax:**

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

- **Example:**

```
SELECT  a.ID, b.NAME as EARNS_HIGHER, a.NAME
as EARNS_LESS, a.SALARY as LOWER_SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY
ORDER BY a.SALARY;
```

LEFT JOIN

- The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.
- **Syntax:**

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

- **Example:**

```
SELECT CUSTOMERS.ID, CUSTOMERS.NAME, ORDERS.DATE, EMPLOYEE.EMPLOYEE_NAME
FROM CUSTOMERS
LEFT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
LEFT JOIN EMPLOYEE
ON ORDERS.OID = EMPLOYEE.EID;
```

ID	NAME	DATE	EMPLOYEE_NAME
1	Ramesh	NULL	NULL
2	Khilan	2009-11-20 00:00:00	REVATHI
3	Kaushik	2009-10-08 00:00:00	ALEKHYA
3	Kaushik	2009-10-08 00:00:00	SARIKA

RIGHT JOIN

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.
- **Syntax:**

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

- **Example:**

```
SELECT ID, NAME, DATE, AMOUNT FROM CUSTOMERS
RIGHT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
WHERE ORDERS.AMOUNT > 1000.00;
```

ID	NAME	DATE	Amount
3	Kaushik	2009-10-08 00:00:00	3000.00
3	Kaushik	2009-10-08 00:00:00	1500.00

FULL JOIN

- The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.
- **Syntax:**

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

- **Example:**

```
SELECT ID, NAME, DATE, AMOUNT FROM CUSTOMERS
  FULL JOIN ORDERS
  ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
  WHERE ORDERS.AMOUNT > 2000.00;
```

ID	NAME	DATE	AMOUNT
3	Kaushik	2009-10-08 00:00:00	3000.00
4	Chaitali	2008-05-20 00:00:00	2060.00

CROSS JOIN

- The CROSS JOIN keyword returns all records from both tables (table1 and table2).
- **Syntax:**

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

- **Example:**

```
SELECT ID, NAME, AMOUNT, DATE, ORDER_RANGE
  FROM CUSTOMERS
  CROSS JOIN ORDERS
  CROSS JOIN ORDER_RANGE;
```

ID	NAME	AMOUNT	DATE	ORDER_RANGE
2	Khilan	1560	2009-11-20 00:00:00	1-100
1	Ramesh	1560	2009-11-20 00:00:00	1-100
2	Khilan	1500.00	2009-10-08 00:00:00	1-100
1	Ramesh	1500.00	2009-10-08 00:00:00	1-100