






# SQL/DBMS

- SQL (Structured Query Language) is a standard language used to manage and manipulate databases.
- It is a DBMS (Database Management System)
- It is a relational database which organises data into tables with rows and columns

## Database Management System (or DBMS)

Software that enables users to interact with databases      an interface to define, create, and manage databases

### Relational Database

organizes data into tables with **rows** and **columns**

**Table** → Entity

**Row** → Record  
or an instance of that entity

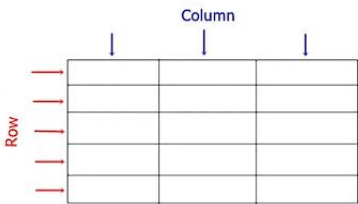
**Column** → Attributes  
or properties of that entity

- To create Database

```
CREATE DATABASE databasename;
```

- To create Table in a database

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




SUBSCRIBE

## Database Concepts

- Database: A collection of related data organized in a structured format
- Table: A collection of rows and columns in a database.
- Row (Record): A single entry in a table.
- Column (Field): A specific attribute of the data.

## Keys

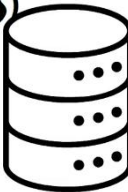
### database management system (or DBMS)




special labels that help identify each row in a table

one or more attributes

**"key value"**





Primary Key, Super Key, Candidate Key, Alternate Key, Foreign Key, Composite Key, and Unique Key

SUBSCRIBE

## 1. Primary Key

### 1. Primary Key

special ID for each row in a table  
identify each record "uniquely"

no two rows in a table  
can have the same values  
for the primary key  
**"distinct identity"**

establish relationships between  
tables, allowing us to  
connect related data together

**"Employees" table**  
"Employee ID" "First Name"  
"Last Name"

**1**

cannot be empty (null)  
or have duplicate values  
**unique & non-empty**

SUBSCRIBE

## 2. Super Key

### 2. Super Key

uniquely identify each record in a table  
"extra information beyond what is necessary"

Employee ID	Employee Name	Employee Address	Employee Designation	Department	Employee Email

super key = employee ID + email address

**primary key is a minimal super key**  
smallest possible subset of attributes  
or columns

**unique + non-null**

**super key is a superset of a primary key**  
super key may contain redundant or extraneous  
attributes or columns that are NOT necessary for uniqueness

SUBSCRIBE

## 3. Unique Key

### 7. Unique Key

each value is unique and not repeated

prevents duplicate entries

Student ID	Class 1	Class 2	Class 3
90001	Biology	Chemistry	Physics
90002	Biology	Physics	Math
90003	Chemistry	Physics	Math
90004	Chemistry	Biology	Physics
90005	Chemistry	Biology	Math

student ID column as a unique key

unique key ensures  
uniqueness of values

(multiple unique keys)

(only one primary key)

SUBSCRIBE



#### 4. Candidate Key

### 3. Candidate Key

used to uniquely identify each record in a table, just like a primary key  
it has the potential to become the primary key

"alternative choices for primary key"

uniqueness & non-null values

all candidate keys are super keys, but not all super keys are candidate keys

"uniqueness"

super keys may include redundant or extraneous attributes that might not comply for uniqueness

"ISBN", "Title", "Author"

ISBN & Title can serve as super keys  
uniqueness & non-nullity constraints

(multiple candidate keys) (only one primary key)

SUBSCRIBE

#### 5. Alternate Key

### 4. Alternate Key

"backup key" that can also identify records uniquely

alternate keys vs candidate keys

candidate keys are eligible to be chosen as the primary key  
alternate keys are additional options that are NOT selected

candidate keys are the primary key contenders, whereas alternate keys are the backup options for uniquely identifying records

all candidate keys can be considered as alternate keys, but not all alternate keys can be considered as candidate keys

primary key could be "Student ID" "Email Address"

(multiple alternate keys) (only one primary key)

SUBSCRIBE

#### 6. Composite Key

### 6. Composite Key

combination of two or more columns that, when taken together, uniquely identifies each record in a table

unique identification

combination of multiple attributes creates a unique identifier for each record

no two rows have the same values for the combined columns

maintain data integrity and provide reliability

first name & last name

SUBSCRIBE

#### 7. Foreign Key

### 5. Foreign Key

way to establish a relationship between two tables

"bridge"

refers to or is used as the primary key column in another table

data integrity and consistency exist in primary key column of referenced table

"establish relationships between tables"

integrity, preventing inconsistencies & accurately representing table relationships

ORDER TABLE:

Order ID	Customer ID	Order Date	Order Value
1	11	24-06-2023	\$44
2	22	25-06-2023	\$52
3	33	26-06-2023	\$8
4	44	27-06-2023	\$68

CUSTOMER TABLE:

Customer Name	Customer Email	Customer ID
Amar	amar@xyz.com	11
Mansi	mansi@xyz1.com	22
Harshu	harshu@xyz2.com	33

SUBSCRIBE

## Index

- A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional storage and maintenance overhead
- Indexes are crucial for enhancing the performance of queries, especially in large databases

### How Indexes Work

Indexes function similarly to an index in a book, allowing the database engine to quickly locate rows without scanning the entire table. Here's how they work in different scenarios:

- B-tree Indexes: Most common type, where indexes are stored in a balanced tree structure, making search operations efficient (logarithmic time complexity).
- Hash Indexes: Use a hash function to map keys to locations, ideal for exact match lookups but not suitable for range queries.
- Bitmap Indexes: Use bitmaps and are efficient for low-cardinality columns in read-heavy environments.

### Advantages of Indexes

- Faster Query Performance: Significantly speeds up data retrieval operations.
- Efficient Sorting: Indexes can make ORDER BY operations faster. Create indexes on columns that are frequently used in WHERE clauses, JOIN conditions, and ORDER BY clauses
- Improved Join Operations: Speeds up joins by providing quick access paths to rows.

### Disadvantages of Indexes

- Storage Overhead: Indexes consume additional disk space.
- Maintenance Overhead: Inserts, updates, and deletes can be slower because the index must be updated.
- Complexity: Over-indexing can lead to performance degradation and complexity in query optimization.

### Types of Index

- Primary - An index on a primary key field which is unique and not null.
- Secondary - An index on non-primary key columns, allowing fast retrieval based on non-primary attributes.
- Composite Index - An index on multiple columns.
- Clustered Index - Determines the physical order of data in a table. There can be only one clustered index per table
- Full Text Index - Special index type optimized for searching text-based columns.
- Unique, Non clustered

Eg:

## Example

sql

Copy code

```
-- Creating a primary index (automatically created when defining primary key)
CREATE TABLE users (
  id INT PRIMARY KEY,
  username VARCHAR(50),
  email VARCHAR(100)
);

-- Creating a secondary index
CREATE INDEX idx_username ON users (username);

-- Creating a composite index
CREATE INDEX idx_name ON persons (last_name, first_name);
```

## SQL Commands

- **SELECT:** Retrieves data from a database.

sql

Copy code

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

- **INSERT:** Adds new rows to a table.

sql

Copy code

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



- **UPDATE:** Modifies existing rows in a table.

sql

Copy code

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

- **DELETE:** Removes rows from a table.

sql

Copy code

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

Note: In above CRUD operations, no need to mention the keyword 'TABLE', just mention the table\_name

But in below Create, Alter, Drop, mention the keyword 'TABLE'

#### Data Definition Language (DDL)

- **CREATE TABLE:** Creates a new table.

sql

Copy code

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

- **ALTER TABLE:** Modifies an existing table.

sql

Copy code

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

- **DROP TABLE:** Deletes a table.

sql

Copy code

```
DROP TABLE table_name;
```

## JOINS

2014.5.18

- **INNER JOIN:** Returns records that have matching values in both tables.

sql

Copy code

```
SELECT columns FROM table1 INNER JOIN table2 ON table1.column = table2.column;
```

- **LEFT JOIN (or LEFT OUTER JOIN):** Returns all records from the left table, and the matched records from the right table.

sql

Copy code

```
SELECT columns FROM table1 LEFT JOIN table2 ON table1.column = table2.column;
```

- **RIGHT JOIN (or RIGHT OUTER JOIN):** Returns all records from the right table, and the matched records from the left table.

sql

Copy code

```
SELECT columns FROM table1 RIGHT JOIN table2 ON table1.column = table2.column;
```

- **FULL JOIN (or FULL OUTER JOIN):** Returns all records when there is a match in either left or right table.

sql

Copy code

```
SELECT columns FROM table1 FULL JOIN table2 ON table1.column = table2.column;
```



## Other Queries

Consider emp table

Query 1 employeesTable

Limit to 1000 rows

```

14 • INSERT INTO emp (EID, ENAME, EADD, ESAL, EDEP, EDATE) VALUES (40, 'anish', 'mango', 18000, 'marketing', '1999-08
15 • INSERT INTO emp (EID, ENAME, EADD, ESAL, EDEP, EDATE) VALUES (20, 'ankush', 'dimna', 17000, 'retail', '2019-11-0
16 • INSERT INTO emp (EID, ENAME, EADD, ESAL, EDEP, EDATE) VALUES (50, 'neha', 'noida', 21000, 'sales', '2001-03-01')
17 • INSERT INTO emp (EID, ENAME, EADD, ESAL, EDEP, EDATE) VALUES (60, 'kapil', 'mumbai', 16000, 'retail', '1995-11-09
18 • INSERT INTO emp (EID, ENAME, EADD, ESAL, EDEP, EDATE) VALUES (30, 'suman', 'dimna', 11000, 'sales', '2000-12-09'
19
20 • ALTER TABLE employees RENAME TO emp;
21 • select * from emp;
??

```

Result Grid

	EID	ENAME	EADD	ESAL	EDEP	EDATE
10	ankit	mango	20000.00	marketing	2021-01-01	
20	ankush	dimna	17000.00	sales	2019-11-03	
30	rohit	sakchi	13000.00	retail	2020-06-24	
40	altaf	goa	23000.00	sales	2018-05-05	
40	anish	mango	18000.00	marketing	1999-08-05	
20	ankush	dimna	17000.00	retail	2019-11-03	
50	neha	noida	21000.00	sales	2001-03-01	
60	kapil	mumbai	16000.00	retail	1995-11-09	
30	suman	dimna	11000.00	sales	2000-12-09	

Result Grid

Form Editor



## ORDER By

- Sorts the result set in ascending or descending order.

```

41
42 #order by clause
43 • select * from emp order by ename;
44 • select * from emp order by eid asc, esal desc; #first sort by eid
45 • select * from emp order by esal desc, eid asc; #first sort by esal

```

	EID	ENAME	EADD	ESAL	EDEP	EDATE
▶	40	ataf	goa	23000.00	sales	2018-05-05
	50	neha	noida	21000.00	sales	2001-03-01
	10	ankit	mango	20000.00	marketing	2021-01-01
	40	anish	mango	18000.00	marketing	1999-08-05
	20	ankush	dimna	17000.00	sales	2019-11-03
	20	ankush	dimna	17000.00	retail	2019-11-03
	60	kapil	mumbai	16000.00	retail	1995-11-09
	30	rohit	sakchi	13000.00	retail	2020-06-24
	30	suman	dimna	11000.00	sales	2000-12-09

## ➤ GROUP BY

- Groups rows that have the same values in specified columns into summary rows. (Note: if u group by edep, then select edep alone or aggregate func)

```

47 #group by
48 • select edep from emp group by edep; #only unique
49 • select edep, count(*) from emp group by edep;
50

```

edep	count(*)
▶ marketing	2
sales	4
retail	3

- You can add condition too wrt to other field/column like print edep who have salary greater than 17000

```

50 • select edep, count(*) from emp where esal > 17000 group by edep;
51

```

edep	count(*)
▶ marketing	2
sales	2

- Department wise get the max/min salary and get the sum/average salaries of the corresponding departments

```

51 #Department wise get the max/min salary and get the sum/average salaries of the corresponding departments
52 • select edep, max(esal), min(esal), sum(esal), avg(esal) from emp group by edep;

```

edep	max(esal)	min(esal)	sum(esal)	avg(esal)
▶ marketing	20000.00	18000.00	38000.00	19000.000000
sales	23000.00	11000.00	72000.00	18000.000000
retail	17000.00	13000.00	46000.00	15333.333333

## ➤ HAVING



- It is always used with 'GROUP By' clause (need that particular column to 'GROUP by' first, then only you can add 'HAVING' to it)
- It is used to restrict the group (similar to 'where')
- Order of execution is WHERE, GROUP by, HAVING, ORDER by
- Eg: Instead of where, using having by in above above statement

```
54 #having by
55 • select edep, count(*) from emp group by edep,esal having esal>17000 ;
```

Result Grid		
Filter Rows:	Export:	Wrap Cell Content:
edep	count(*)	
marketing	1	
sales	1	
marketing	1	
sales	1	

```
56 • select edep, esal, count(*) from emp group by edep,esal having esal>17000 ;
```

Result Grid		
Filter Rows:	Export:	Wrap Cell Content:
edep	esal	count(*)
marketing	20000.00	1
sales	23000.00	1
marketing	18000.00	1
sales	21000.00	1

- To get simple salary range and also create a duplicate table with same structure

```
46 #to find details of employees who have salary as 23000,11000 and 13000
47 • select * from emp where esal in (23000,11000,13000);
48 #to find details of employees who have salary between 10000 and 15000
49 • select * from emp where esal between 10000 and 15000;
50
51 #to create a duplicate table with same structure
52 • create table emp2 as select * from emp;
53
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	EID	ENAME	EADD	ESAL	EDEP	EDATE
▶	30	rohit	sakchi	13000.00	retail	2020-06-24
	30	suman	dimna	11000.00	sales	2000-12-09

- To find patterns

```

39  #to find all details of employees whose name starts with 'a'
40 • select * from emp where ename like 'a%';
41  #to find all details of employees whose name ends with 'a'
42 • select * from emp where ename like '%a';
43  #to find all details of employees whose name does not have letter 'a'
44 • select * from emp where ename not like '%a%';
45

```

Result Grid						
Filter Rows: <input type="text"/>						
Export: <input type="button" value=""/>						
Wrap Cell Content: <input type="button" value=""/>						
EID	ENAME	EADD	ESAL	EDEP	EDATE	
30	rohit	sakchi	13000.00	retail	2020-06-24	

### ➤ To find max salary, 2nd max salary or nth max salary

```

51  #to find max salary
52 • select * from emp where esal = (select max(esal) from emp);
53
54  #to find 2nd max salary
55 • select max(esal) from emp where esal < (select max(esal) from emp);
56  #or
57 • select max(esal) from emp where esal not in (select max(esal) from emp);
58
59  #to find nth max salary (offset = n-1)
60 • select distinct esal from emp order by esal desc limit 1 offset 2;

```

Result Grid						
Filter Rows: <input type="text"/>						
Export: <input type="button" value=""/>						
Wrap Cell Content: <input type="button" value=""/>						
Fetch rows: <input type="button" value=""/>						
esal						
20000.00						

### ➤ To find repeating/duplicate values of a column like eid or ename

```

62  #to find repeating/duplicate values of a column like eid or ename
63 • select eid, count(*) from emp group by eid having count(*) > 1;

```

Result Grid						
Filter Rows: <input type="text"/>						
Export: <input type="button" value=""/>						
Wrap Cell Content: <input type="button" value=""/>						
eid	count(*)					
20	2					
30	2					
40	2					

### ➤ To find other patterns