SQL FUNDAMENTALS

AGENDA

- Introduction Database
- Introduction to SQL
- Difference between Database and Spreadsheet
- What are Data types in SQL
- SQL Data types
- SQL Constraints
- Types of SQL Commands
- DDL, DML, DQL, TCL
- SQL Functions
- SQL Joins
- SQL sub queries

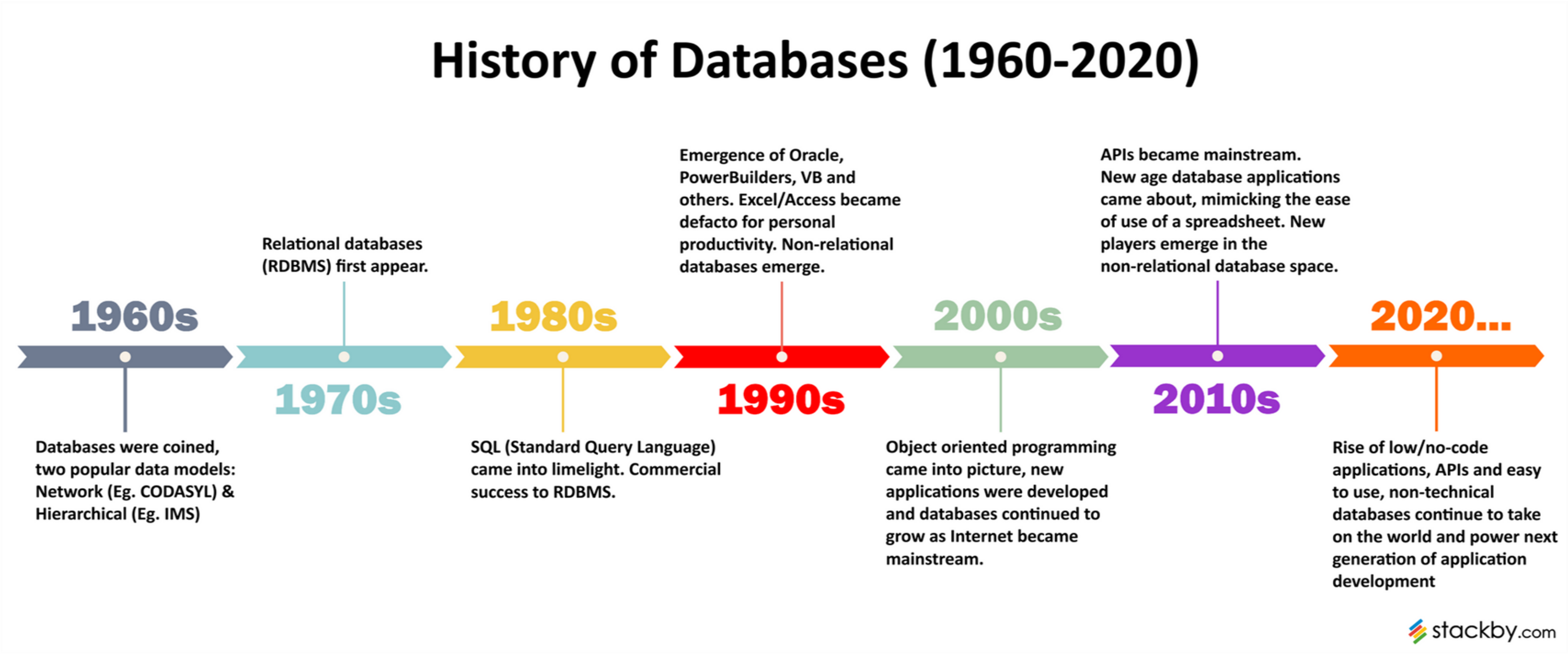
WHAT IS DATA

- Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.
- Word 'Data' is originated from the word 'datum' that means 'single piece of information.' It is plural of the word datum.
- In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

WHAT IS DATABASE

- A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a **database management system** (DBMS).
- You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.
- The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data.
- There are many dynamic websites on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.
- There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

EVOLUTION OF DATABASE



History of Databases (1960-2020)

1960s

Databases were coined, two popular data models: Network (Eg. CODASYL) & Hierarchical (Eg. IMS)

1970s

Relational databases (RDBMS) first appear.

1980s

SQL (Standard Query Language) came into limelight. Commercial success to RDBMS.

1990s

Emergence of Oracle, PowerBuilders, VB and others. Excel/Access became defacto for personal productivity. Non-relational databases emerge.

2000s

Object oriented programming came into picture, new applications were developed and databases continued to grow as Internet became mainstream.

2010s

APIs became mainstream. New age database applications came about, mimicking the ease of use of a spreadsheet. New players emerge in the non-relational database space.

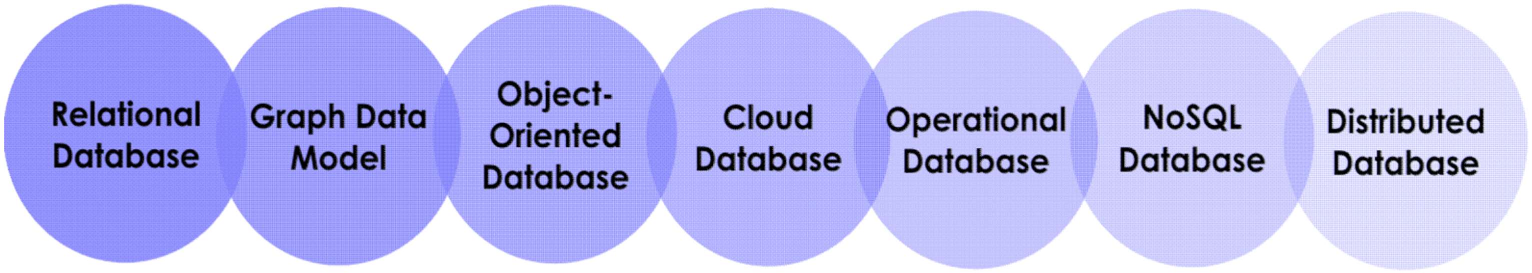
2020...

Rise of low/no-code applications, APIs and easy to use, non-technical databases continue to take on the world and power next generation of application development

stackby.com

TYPES OF DATABASES

Different types of databases:



Relational Database

Graph Data Model

Object-Oriented Database

Cloud Database

Operational Database

NoSQL Database

Distributed Database

INTRODUCTION TO SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- Data analysts and developers learn and use SQL because it integrates well with different programming languages.

DIFFERENCE BETWEEN DATABASE AND SPREADSHEET

- Databases and spreadsheets (such as Microsoft Excel) are both convenient ways to store information. The primary differences between the two are:
  - How the data is stored and manipulated
  - Who can access the data
  - How much data can be stored
- Spreadsheets were originally designed for one user, and their characteristics reflect that. They're great for a single user or small number of users who don't need to do a lot of incredibly complicated data manipulation. Databases, on the other hand, are designed to hold much larger collections of organized information—massive amounts, sometimes.
- Databases allow multiple users at the same time to quickly and securely access and query the data using highly complex logic and language.

WHAT IS DATA TYPE IN SQL

- The data type of a column defines what value the column can hold: integer, character, date and time, binary, and so on.
- Each column in a database table is required to have a name and a data type.
- An SQL developer must decide what type of data that will be stored inside each column when creating a table. The data type is 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.

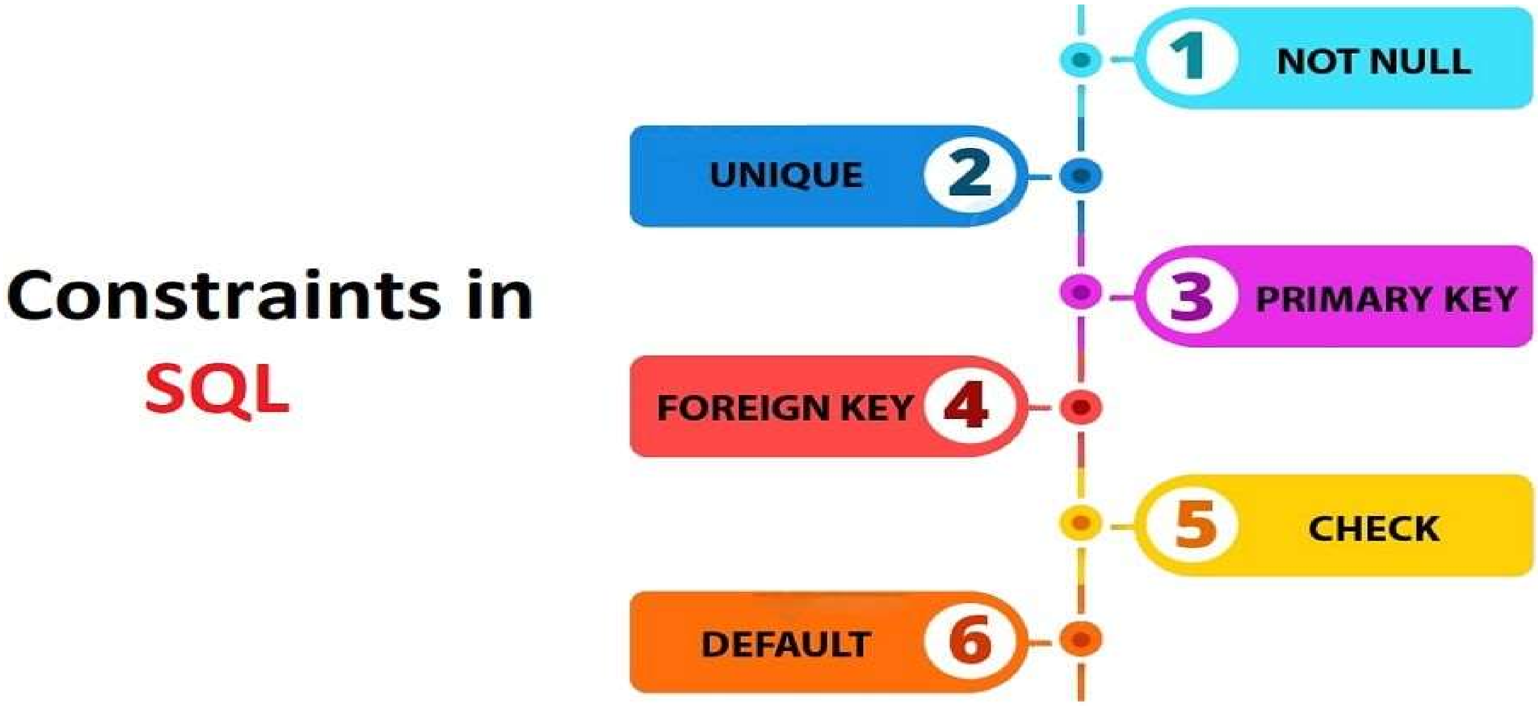
DIFFERENT DATA TYPES

Data Type	Description
INT	Stores numeric values in the range of -2147483648 to 2147483647
DECIMAL	Stores decimal values with exact precision.
CHAR	Stores fixed-length strings with a maximum size of 255 characters.
VARCHAR	Stores variable-length strings with a maximum size of 65,535 characters.
TEXT	Stores strings with a maximum size of 65,535 characters.
DATE	Stores date values in the YYYY-MM-DD format.
DATETIME	Stores combined date/time values in the YYYY-MM-DD HH:MM:SS format.

SQL CONSTRAINTS

- SQL constraints are a set of rules implemented on tables in relational databases to dictate what data can be inserted, updated or deleted in its tables. This is done to ensure the accuracy and the reliability of information stored in the table.
- Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table.

TYPES OF SQL CONSTRAINTS



Constraints in SQL

1 NOT NULL

2 UNIQUE

3 PRIMARY KEY

4 FOREIGN KEY

5 CHECK

6 DEFAULT

SQL CONSTRAINTS

NOT NULL - The NOT NULL constraint specifies that the column does not accept NULL values. This means if NOT NULL constraint is applied on a column then you cannot insert a new row in the table without adding a non-NULL value for that column.

UNIQUE - The UNIQUE constraint restricts one or more columns to contain unique values within a table.

PRIMARY KEY - The PRIMARY KEY constraint identify the column or set of columns that have values that uniquely identify a row in a table. No two rows in a table can have the same primary key value. Also, you cannot enter NULL value in a primary key column.

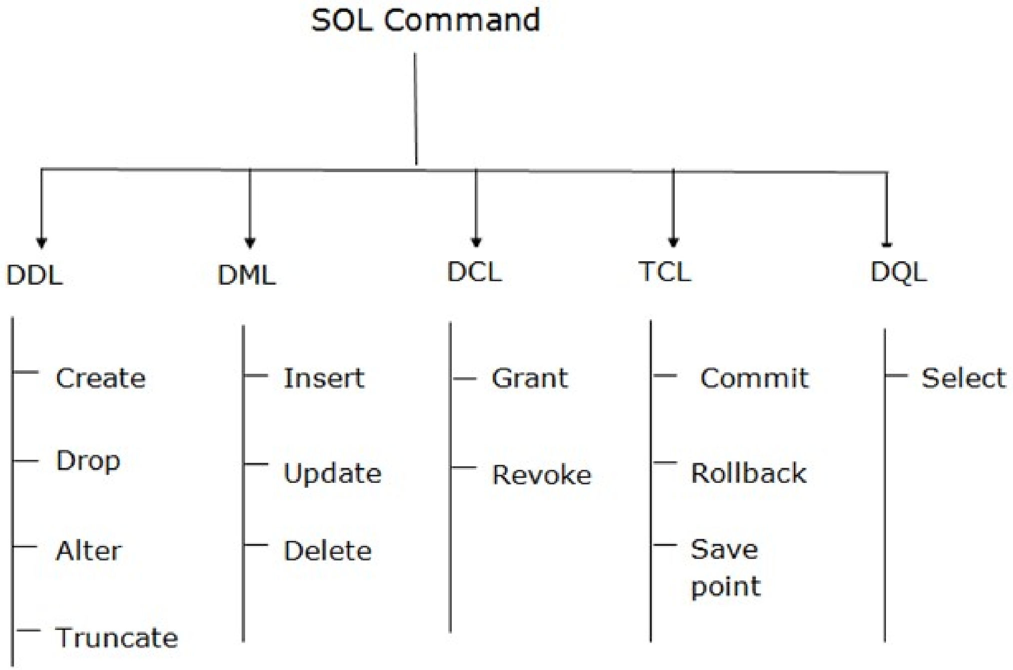
FOREIGN KEY - constraints in MySQL are used to enforce referential integrity between tables. A foreign key in one table points to a primary key in another table, ensuring that the relationship between the two tables remains consistent.

CHECK- The CHECK constraint in MySQL is used to ensure that all values in a column satisfy a specific condition.

DEFAULT - The DEFAULT constraint specifies the default value for the columns.

TYPES OF SQL COMMANDS

The following is the list of five widely used SQL Commands.



SOL Command

DDL

- Create
- Drop
- Alter
- Truncate

DML

- Insert
- Update
- Delete

DCL

- Grant
- Revoke

TCL

- Commit
- Rollback
- Save point

DQL

- Select

DDL

- DDL stands for data definition language. DDL Commands deal with the schema, i.e., the tables in which our data is stored.
- All the structural changes such as creation, deletion and alteration on the table can be carried with the DDL commands in SQL.
- Commands covered under DDL are:
  - **CREATE**
  - **ALTER**
  - **DROP**
  - **TRUNCATE**
  - **RENAME**

HANDS-ON EXAMPLE

- Create a database named sql_exercises
- Create a table employees
- Alter the table employees by adding one column
- Rename the employees table
- Truncate the table
- Drop the table from database

DML

- It stands for Data Manipulation Language. The DML commands deal with the manipulation of existing records of a database. It is responsible for all changes that occur in the database.
- The changes made in the database using this command can't save permanently because its commands are not auto-committed. Therefore, changes can be rollback.
- The following commands come under DML language:

- **INSERT**: It is a SQL query that allows us to add data into a table's row.
- **UPDATE**: This command is used to alter or modify the contents of a table.
- **DELETE**: This command is used to delete records from a database table, either individually or in groups.

HANDS-ON EXAMPLE

- Insert some data into the table
- Update the data for any entry
  - Turn off the safe mode first. Uncheck-Edit>Preferences>Sql editor>Safe Updates
  - Reconnect the server
- Insert some more data into the table
- Delete one row now

DQL

- It stands for Data Query Language. The DQL commands deal with extracting data from tables. Various SQL clauses/operators can be used along with Select command.
- The following commands come under DQL language:
- **SELECT**: This command is used to extract information from a table.
  - Clauses/operators used along with SELECT command are:
    - From
    - Where
    - Like
    - Order by, Limit, offset
    - Null
    - And, Or, Not

HANDS-ON EXAMPLE

- Selecting employees with first_name starting with 'a', sorted by salary, and limiting results using hr database.
- Selecting employees with salary less than 3000 and hire year is 2000

TCL

- It stands for Transaction Control Language. The TCL commands deal controlling the transactions.
- The following commands come under TCL language:
- **COMMIT**: To save the work done.
- **ROLL BACK**: Restore database to original state since the last COMMIT.

HANDS-ON EXAMPLE

- We will change the salary of a particular employee in hr database and then check the use of ROLLBACK and COMMIT

DCL

- DCL (Data Control Language) is a subset of commands used to control access to data within a database.
- The following commands come under CL language:
- GRANT : This command is used to give users access privileges to the database objects. It can be used to provide specific permissions to users, such as the ability to SELECT, INSERT, UPDATE, DELETE, or execute specific functions and procedures.
- REVOKE : This command is used to remove previously granted privileges from users.

EXAMPLE

Grant SELECT privilege to hr_user:

```
GRANT SELECT ON hr.employees TO 'hr_user'@'localhost';
```

Verify the granted privileges:

```
SHOW GRANTS FOR 'hr_user'@'localhost';
```

Revoke SELECT privilege from hr_user:

```
REVOKE SELECT ON hr.employees FROM 'hr_user'@'localhost';
```

Verify the revoked privileges:

```
SHOW GRANTS FOR 'hr_user'@'localhost';
```

Explanation:

- The GRANT statement is used to assign specific privileges (like SELECT, INSERT, UPDATE, DELETE, etc.) to a user on a specific database object (table, database, etc.).
- The REVOKE statement is used to remove previously granted privileges from a user.

STATEMENTS VS CLAUSES VS FUNCTIONS

SQL Statements:

- SQL statements are complete, standalone commands that perform specific actions in the database.
- Examples of SQL statements include SELECT , INSERT , UPDATE , DELETE etc.

SQL Clauses:

- Clauses are components of SQL statements that provide additional instructions or conditions.
- They are used to filter, sort, or group data, among other things, within a SQL statement.
- Examples of SQL clauses include WHERE , ORDER BY , GROUP BY ,JOIN etc.

SQL Functions:

- SQL functions are operations or calculations applied to data values.
- They can be used within SQL statements to perform various tasks, such as mathematical operations, string manipulations, date calculations, and aggregate calculations.
- Examples of SQL functions include COUNT(), SUM(), AVG(), MAX(), UPPER() etc.

SQL FUNCTIONS

MySQL Numeric Functions

ABS	Returns the absolute value of a number
AVG	Returns the average value of an expression
CEIL	Returns the smallest integer value that is >= to a number
COUNT	Returns the number of records returned by a select query
FLOOR	Returns the largest integer value that is <= to a number
MAX	Returns the maximum value in a set of values

MySQL String Functions

CONCAT	Adds two or more expressions together
INSERT	Inserts a string within a string at the specified position and for a certain number of characters
LENGTH	Returns the length of a string (in bytes)
LOWER	Converts a string to lower-case
REPLACE	Replaces all occurrences of a substring within a string, with a new substring
SUBSTR	Extracts a substring from a string (starting at any position)
TRIM	Removes leading and trailing spaces from a string

SQL FUNCTIONS

<u>MySQL Date Functions</u>

ADDDATE	Adds a time/date interval to a date and then returns the date
ADDTIME	Adds a time interval to a time/datetime and then returns the time/datetime
CURRENT_DATE	Returns the current date
CURTIME	Returns the current time
MONTH	Returns the month part for a given date
WEEK	Returns the week number for a given date
YEAR	Returns the year part for a given date

<u>MySQL Advanced Functions</u>

CASE	Goes through conditions and return a value when the first condition is met
CAST	Converts a value (of any type) into a specified datatype
COALESCE	Returns the first non-null value in a list
IF	Returns a value if a condition is TRUE, or another value if a condition is FALSE
IFNULL	Return a specified value if the expression is NULL, otherwise return the expression
ISNULL	Returns 1 or 0 depending on whether an expression is NULL

HANDS-ON EXAMPLE

- We will write queries using various functions :
  - Numeric Functions
  - String Functions
  - Date Functions
  - Advanced Functions

SQL CLAUSES IN DQL

- Various clauses used in SQL for querying as per Order of execution are:
- 1. FROM
- 2. ON
- 3. JOIN
- 4. WHERE
- 5. GROUP BY
- 6. HAVING
- 7. SELECT
- 8. DISTINCT
- 9. ORDER BY/LIMIT

HANDS ON DQL

- Perform the queries to fetch data from “hr” Database.
- We will see the use of various functions/clauses to extract the required data.
  - Wild cards
  - WHERE and LIKE
  - DISTINCT
  - GROUP BY and HAVING
  - ORDER BY , DESC, LIMIT
  - IF, CASE , WHEN , THEN, ELSE, END

ADVANCED SQL QUERIES

SQL JOINS

Inner Join

Table 1

ROWS1	ENTRY1
1	A
1	B
2	A
3	NULL
NULL	C
NULL	NULL
4	D

Table 2

ROWS1	ENTRY2
1	X
1	X
2	Y
3	Z
4	X
NULL	M
4	NULL
5	NULL
5	M

Inner Join

ROWS1	ENTRY1	ENTRY2
1	B	X
1	A	X
1	B	X
1	A	X
2	A	Y
3	NULL	Z
4	D	X
4	D	NULL

Left Join

Table 1

ROWS1	ENTRY1
1	A
1	B
2	A
3	NULL
NULL	C
NULL	NULL
4	D

Table 2

ROWS1	ENTRY2
1	X
1	X
2	Y
3	Z
4	X
NULL	M
4	NULL
5	NULL
5	M

LEFT Join

ROWS1	ENTRY1	ENTRY2
1	A	X
1	A	X
1	B	X
1	B	X
2	A	Y
3	NULL	Z
NULL	C	NULL
NULL	NULL	NULL
4	D	NULL
4	D	X

Right Join

Table 1

ROWS1	ENTRY1
1	A
1	B
2	A
3	NULL
NULL	C
NULL	NULL
4	D

Table 2

ROWS1	ENTRY2
1	X
1	X
2	Y
3	Z
4	X
NULL	M
4	NULL
5	NULL
5	M

RIGHT Join

ROWS1	ENTRY1	ENTRY2
1	X	B
1	X	A
1	X	B
1	X	A
2	Y	A
3	Z	NULL
4	X	D
NULL	M	NULL
4	NULL	D
5	NULL	NULL
5	M	NULL

Full Outer Join

Table 1

ROWS1	ENTRY1
1	A
1	B
2	A
3	NULL
NULL	C
NULL	NULL
4	D

Table 2

ROWS1	ENTRY2
1	X
1	X
2	Y
3	Z
4	X
NULL	M
4	NULL
5	NULL
5	M

Full Outer Join

ROWS1	ENTRY1	ENTRY2
1	A	X
1	B	X
2	A	Y
3	NULL	Z
NULL	C	NULL
NULL	NULL	NULL
4	D	NULL
4	D	X
1	X	B
1	X	A
2	Y	A
3	Z	NULL
4	X	D
NULL	M	NULL
4	NULL	D
5	NULL	NULL
5	M	NULL

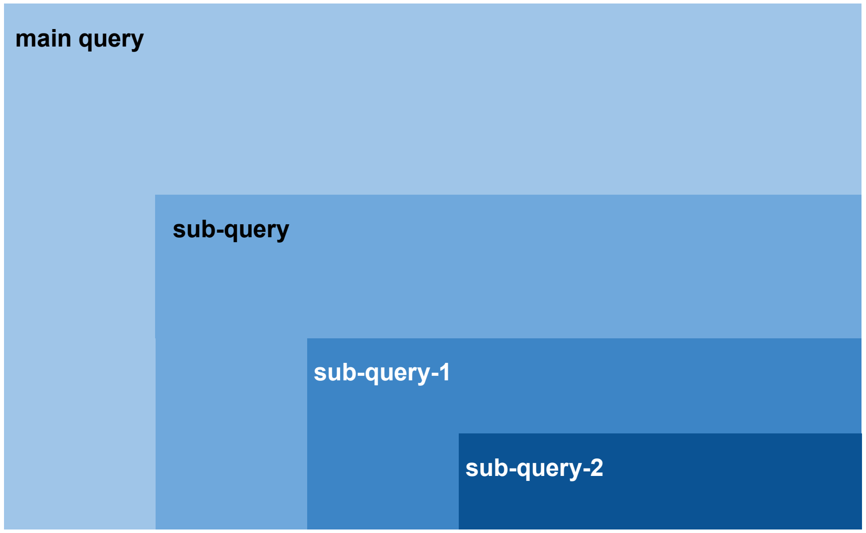
HANDS ON

- Run the various queries using “hr” data base to fetch the data from various tables using joins

NESTED QUERY OR SUB QUERIES

A subquery is a query nested inside another query.

Write queries within queries to find answers to complex questions!



main query

sub-query

sub-query-1

sub-query-2

Examples	Concept
Example-1	Single Row Subquery
Example-2	Multiple Row Subquery
Example-3	Multiple Column Subquery
Example-4	Nested Subquery
Example-5	Subquery using Aggregare Function
Example-6	Subquery with Joins
Example-7	Subquery in FROM Clause
Example-8	Subquery using EXISTS

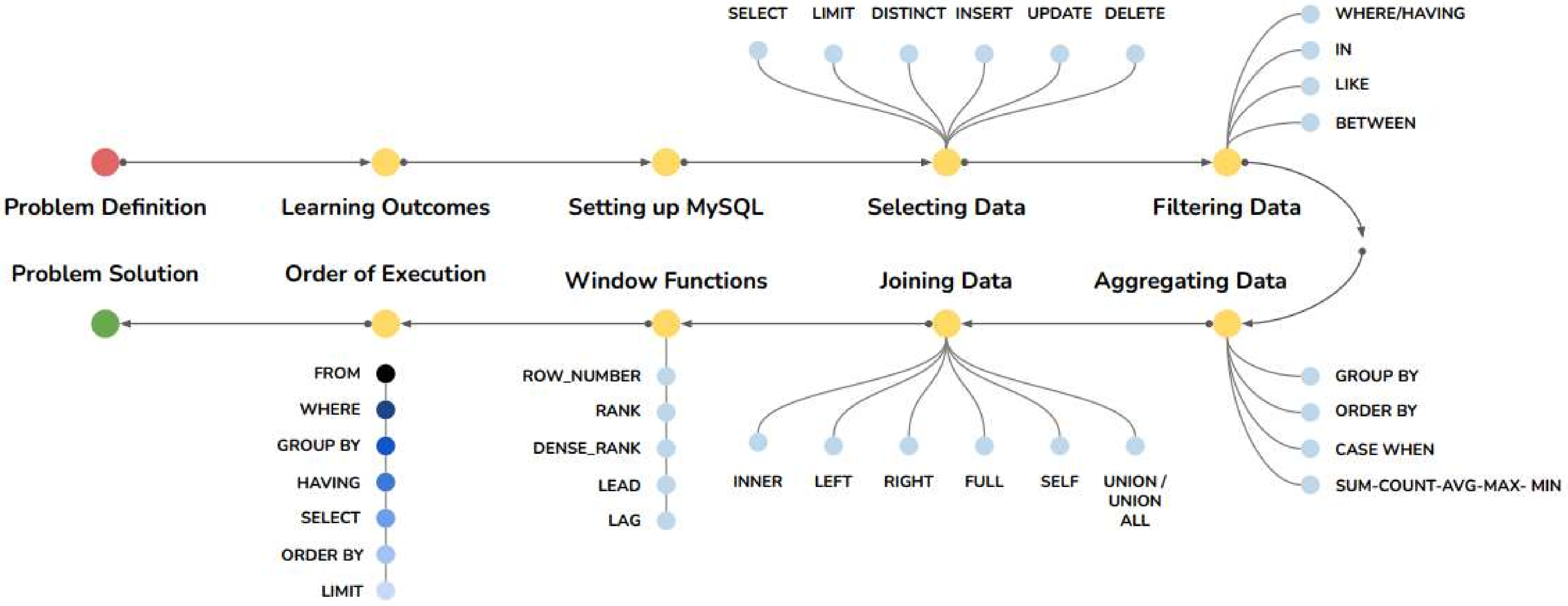
Example of a subquery in the:

- SELECT clause can be used to return a single value for each row selected by the outer query
- WHERE clause can be used to filter results based on a condition that involves a separate query
- FROM clause can be used to create a temporary table that the outer query can then use
- HAVING clause can be used to filter groups based on the results of a separate query
- IN operator can be used to filter results based on a list of values returned by the subquery.
- EXISTS operator can be used to check for the existence of rows returned by the subquery

HANDS ON

- We will write various queries to fetch data from "hr" database
- We will see how to use of Joins and sub queries to fetch data as asked.

SQL SUMMARY


Problem Definition
Learning Outcomes
Setting up MySQL
Selecting Data
SELECT
LIMIT
DISTINCT
INSERT
UPDATE
DELETE
Filtering Data
WHERE/HAVING
IN
LIKE
BETWEEN
Aggregating Data
GROUP BY
ORDER BY
CASE WHEN
SUM-COUNT-AVG-MAX- MIN
Joining Data
INNER
LEFT
RIGHT
FULL
SELF
UNION / UNION ALL
Window Functions
ROW_NUMBER
RANK
DENSE_RANK
LEAD
LAG
Order of Execution
FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY
LIMIT
Problem Solution

INTEGRATING SQL WITH PYTHON

- Integrating SQL with Python provides several benefits and is necessary for various reasons, especially when building applications that require data storage, retrieval, and manipulation. Here are some key reasons why integrating SQL with Python is important:

1. **Data Storage and Management**
   - **Persistent Data Storage**: SQL databases provide a reliable and structured way to store data persistently. Unlike in-memory storage, data in SQL databases remains available even after the application is closed or restarted.
   - **Efficient Data Management**: SQL databases are designed to efficiently handle large volumes of data, providing quick access and modifications through structured queries.

2.Data Manipulation and Retrieval

- **Advanced Querying**: SQL allows complex queries to filter, sort, aggregate, and join data across multiple tables. This is crucial for extracting meaningful information from datasets.
- **Data Integrity and Consistency**: SQL databases enforce data integrity and consistency through constraints, transactions, and relational structures, ensuring the accuracy and reliability of data.

3. Separation of Concerns

- **Modular Architecture**: Integrating SQL with Python separates data management logic from application logic. This modularity makes code easier to maintain, test, and debug.
- **Reusability**: By separating SQL queries and Python code, you can reuse the same database queries across different parts of your application or even different projects.

4. **Scalability and Performance**

   - **Scalability**: SQL databases can handle growing datasets and concurrent users efficiently. Using SQL with Python allows applications to scale without significant performance degradation.
   - **Optimization**: SQL databases come with various optimization features like indexing, caching, and query optimization, which enhance the performance of data operations.

5. **Data Analysis and Reporting**

   - **Analytical Queries**: SQL is powerful for data analysis tasks, allowing you to perform complex calculations, aggregations, and statistical analyses directly in the database.
   - **Reporting**: By integrating SQL with Python, you can easily generate reports, dashboards, and visualizations based on the data stored in your SQL database.

PRACTICAL USE CASES

- **Web Applications**: Store user data, session information, and application settings in a database.
- **Data-Driven Applications**: Build applications that require persistent data, such as inventory management systems, customer relationship management (CRM) systems, and content management systems (CMS).
- **Data Analysis and Machine Learning**: Store and preprocess data before feeding it into machine learning models.
- **Financial Applications**: Manage transactions, account details, and financial records securely and efficiently.

HANDS ON

- We will connect the jupyter notebook with mysql and fetch the data.

THANKS……….