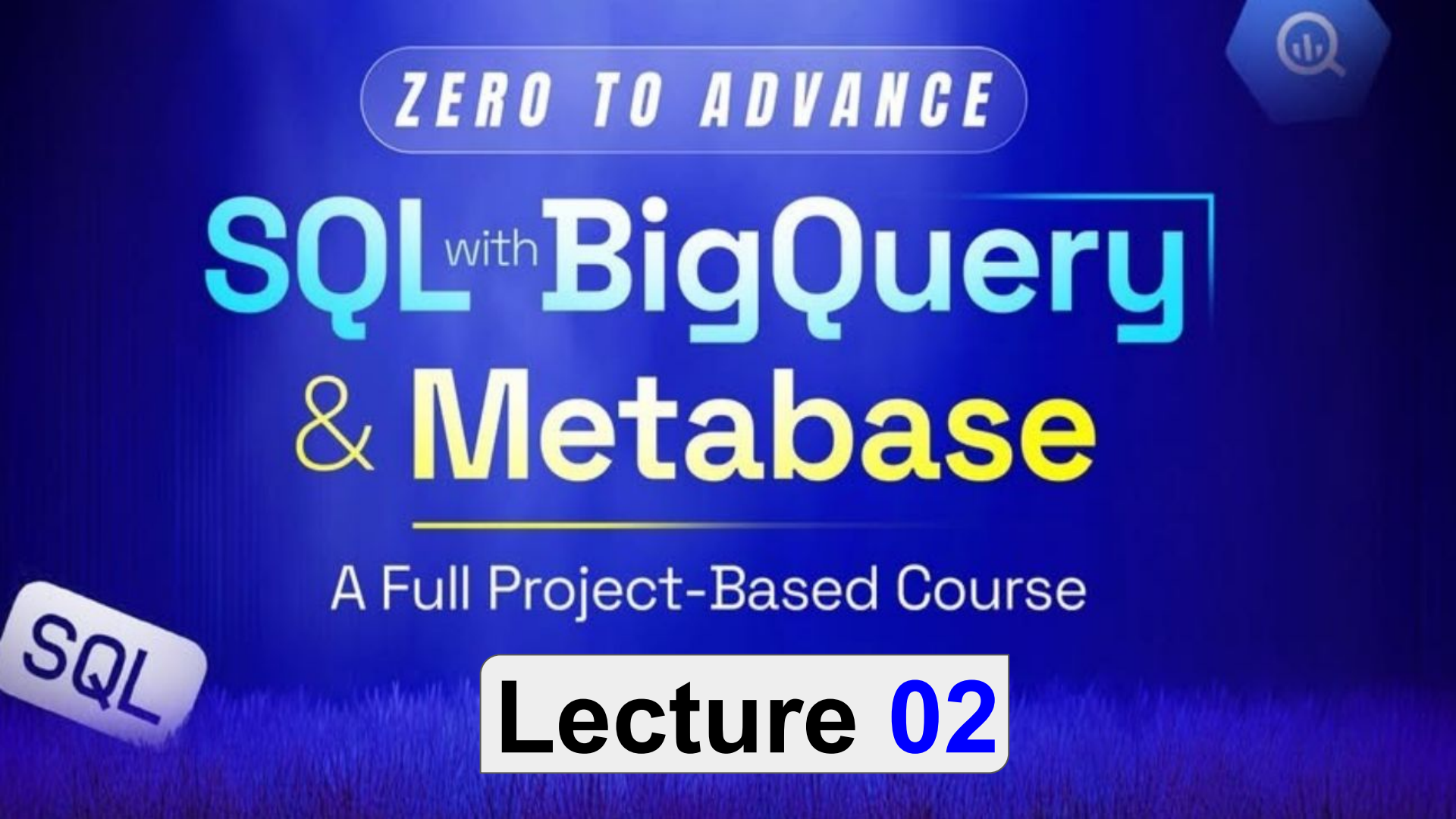

ZERO TO ADVANCE
SQL with BigQuery
& Metabase
A Full Project-Based Course
SQL
Lecture 02

Hello! I Am

SAJIB KHAN

Executive, Data Analytics & Strategy
at **Pathao LTD.**

Support Instructor,
Data Analytics & BI Career Path,

Lead Instructor,
Zero to Advance SQL with Big Query & Metabase
at **Interactive Cares**

sajibkhansk.netlify.app

/in/sajibkhansk

/sajibkhanofficial

+88 01717 015761

sajibkhansk19@gmail.com

Class 2: Database Fundamentals & Write Your First Query

📦 Database

- A **collection of organized data**, typically stored electronically.
- Acts as a container for schemas and tables.

🗂️ Schema

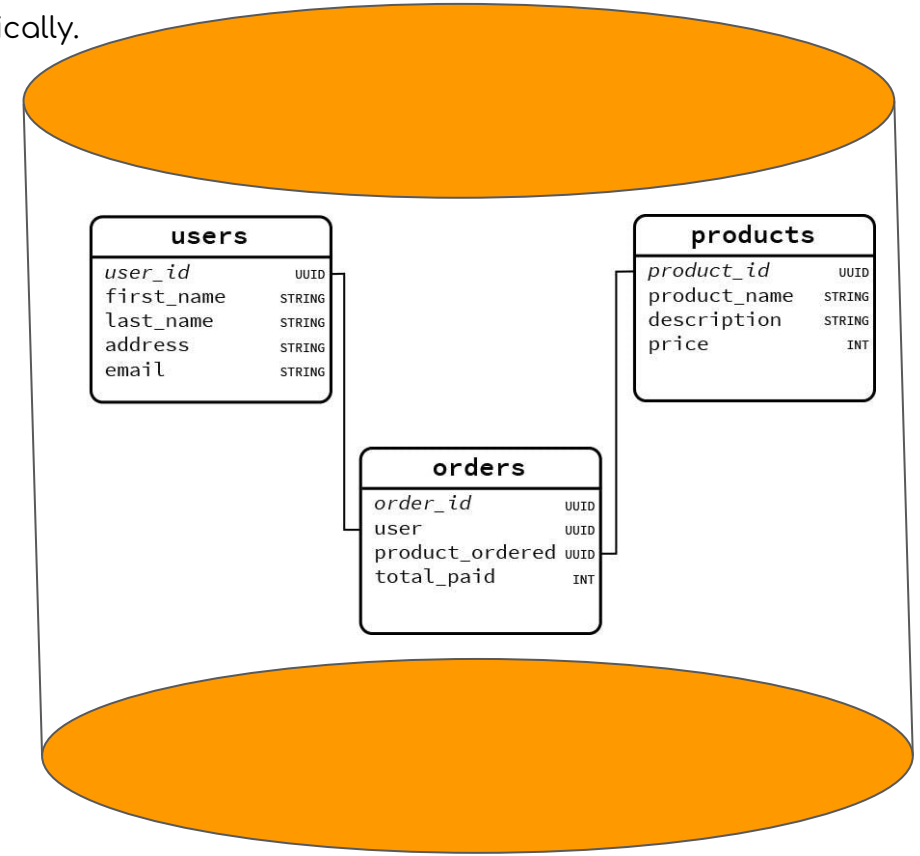
- A **blueprint** of the database.
- Defines **tables, fields, relationships, views, indexes**, etc.

📊 Table

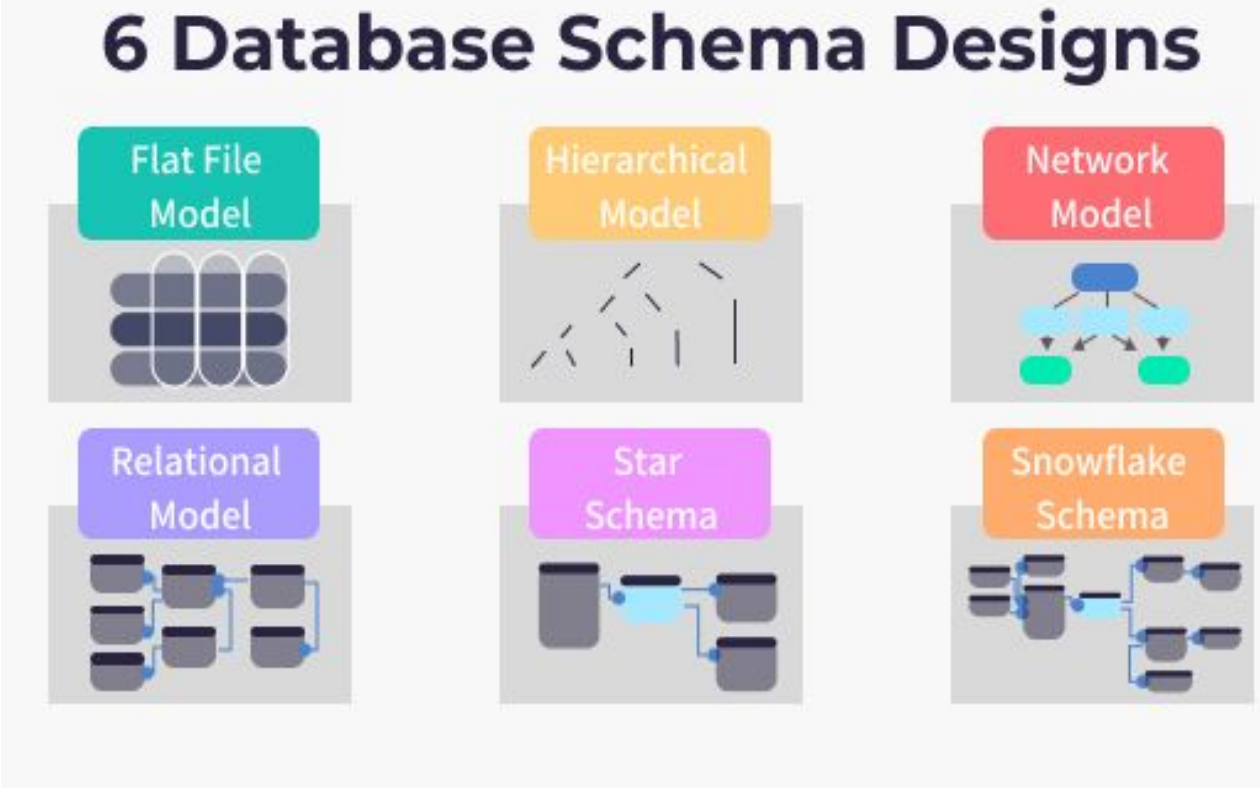
- Made up of **rows (records)** and **columns (fields)**.
- Think of it like an **Excel sheet** – each row is a new entry.

🔤 Field

- A **single column** in a table.
- Represents one type of data (e.g., Name, Age, Email).



Schema



Schema Type	Example	Use Case
Flat File Model	A CSV file with employee data:ID, Name, Department, Salary1, Alice, HR, 500002, Bob, IT, 60000	Simple spreadsheets, log files
Hierarchical Model	Organization chart:CEO → VP → Manager → Employee	XML data, file systems
Network Model	Student ↔ CourseA student can enroll in multiple coursesA course can have many students	Telecommunications, transport systems
Relational Model	Tables:Students (student_id)Courses (course_id)Enrollments (student_id, course_id)	Standard SQL databases (MySQL, PostgreSQL)
Star Schema	Fact Table: Sales(sales_id, date_id, product_id, revenue)Dimension Tables:- Date(date_id, day, month, year)- Product(product_id, name, category)	Data warehousing, reporting
Snowflake Schema	Normalized Dimensions:Product → Product Category → Department	Complex analytical databases

Fact VS Dimension Table

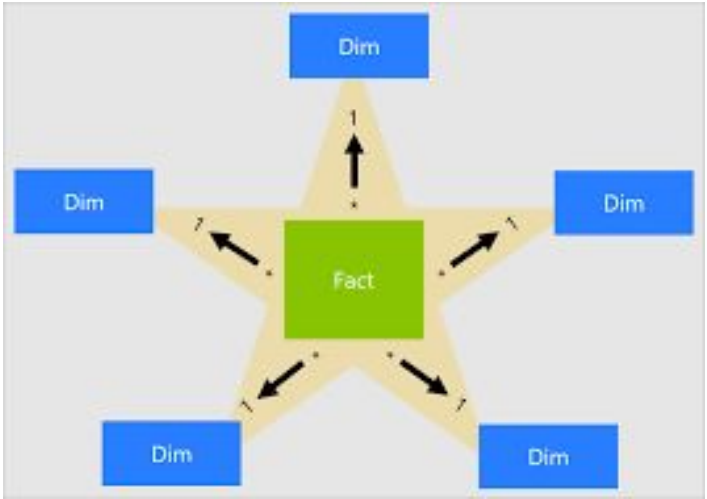
FACT TABLES VS. DIMENSION TABLES

FACT TABLES

- Represent verbs – (hiring people, producing widgets)
- Change constantly (day, minute, even second)
- Long and thin (lots of rows, few columns)
- Associated with aggregation

DIMENSION TABLES

- Represent nouns associated with those actions (employees, customers, countries, dates)
- Change, but much less frequently (months, years or decades)
- Short and wide (fewer rows, more columns)
- Associated with filtering and grouping



Relationships in SQL

Relationships in SQL refer to the **associations** or **connections between tables** in a relational database. These relationships are established using **foreign keys**, which are columns in a table that refer to the primary key in another table. Relationships help organize and structure data, allowing for efficient data **retrieval** and **maintaining** data integrity.

Type of Relationships in SQL

There are different types of relationships: one-to-one, one-to-many, many-to-many, and self-referencing.

Ref

SQL Constraints & Their Types

What are SQL Constraints?

SQL Constraints are **rules** applied to columns in a table to **enforce data integrity, prevent invalid data,** and **maintain consistency** in a database.

Types of SQL Constraints:

1. NOT NULL – Prevents NULL values in a column.
2. UNIQUE – Ensures all values in a column are unique.
3. PRIMARY KEY – Combines NOT NULL + UNIQUE to identify each row uniquely.
4. FOREIGN KEY – Links two tables, ensuring valid relationships.
5. CHECK – Validates values based on a condition.
6. DEFAULT – Assigns a default value if none is provided.

Data Types & Auto Increment in SQL

Date & Time Types

- **DATE** – Stores only date (YYYY-MM-DD).
- **DATETIME** – Stores date and time (YYYY-MM-DD HH:MM:SS).

String Type

- **VARCHAR(n)** – Variable-length string up to *n* characters.

Numeric Types

- **INT** – Integer (whole numbers).
- **NUMBER(p,s)** – Number with precision *(p)* and scale *(s)*. *(Mostly in Oracle)*
- **FLOAT** – Floating point numbers with decimals.

Auto Increment

- Automatically increases the value in a column (usually primary key) for each new row.
- Syntax (MySQL):

Database Design

What is Indexing?

Indexing makes columns faster to query by creating pointers to where data is stored within a database.

Imagine you want to find a piece of information that is within a large database. To get this information out of the database the computer will look through every row until it finds it. If the data you are looking for is towards the very end, this query would take a long time to run.

Ref

Installing MySQL Workbench


ZERO TO ADVANCE
SQL with BigQuery
& Metabase
A Full Project-Based Course
THANK YOU