Detailed Short Notes – Lecture 01 (IT2140)

Database Design and Development
Introduction to DBMS, Database Design Process, and Data-Intensive Applications

---

1. Data vs Information

- **Data**:
  - Raw, unorganized facts (numbers, text, images, symbols).
  - On its own, it does not provide meaning.
  - Example: "100", "John", "2025".
- **Information**:
  - Processed, structured, or organized data.
  - Provides context and meaning.
  - Example: "John scored 100 marks in IT2140 in 2025."

---

2. Database & DBMS

- **Database**:
  - A collection of related data.
  - Organized for easy access, management, and updating.
  - Example: Student records system storing names, IDs, grades.
- **DBMS (Database Management System)**:
  - A software system that helps to **define, construct, manipulate, and share** databases.
  - Examples: MySQL, Oracle, PostgreSQL, MongoDB.
- **Benefits of DBMS**:
  - Centralized data management.
  - Easy sharing among multiple users.
  - Security and backup features.

- Reduces redundancy and inconsistency.

---

3. Database System Environment

The environment consists of **five main components**:

1. **Hardware** – Physical devices (servers, storage, computers).
2. **Software** – DBMS software, applications, OS.
3. **People** – Users (DB administrators, developers, end-users).
4. **Procedures** – Rules, instructions, and processes for database use.
5. **Data** – The actual stored information.

---

4. Database Approach vs File Processing System

- **File Processing System**:
  - Data stored separately in files.
  - Redundant, inconsistent, difficult to maintain.
- **Database Approach**:
  - **Self-describing** – Stores metadata with data.
  - **Data Abstraction** – Programs are insulated from storage details.
  - **Multiple Views** – Different users can see customized views.
  - **Multiuser Transactions** – Supports simultaneous access.

Advantages of Database Approach:

- **Data Independence** – Applications are independent from physical storage.
- **Efficient Access** – Uses indexing and optimization.
- **Integrity Enforcement** – Constraints (e.g., ID must be unique, name must be a string).
- **Security** – Access control for different users.
- **Backup & Recovery** – Protection from system failures.

- **Concurrent Access** – Many users can work simultaneously.

---

5. Database Design

- **Why Important?**
  - Prevents unnecessary/omitted data.
  - Avoids incorrect or inconsistent results.
  - Ensures efficiency in queries.
  - Allows adaptability to user requirements.
  - Saves time and cost in the long run.

---

6. Database Design Process (Six Phases)

1. **Requirement Collection & Analysis**
   - Identify **data requirements** (objects, attributes).
   - Determine **relationships** among objects.
   - Identify **transactions** (operations on data).
   - Consider **constraints** (performance, integrity, security).
   - Sources of info:
     - Documents (forms, reports, guidelines).
     - Interviews with end users.
     - Reviewing existing automated systems.
2. **Conceptual Database Design**
   - Create a **high-level model** (e.g., ER Model).
   - Focus on objects, attributes, and relationships.
3. **Logical Database Design**
   - Choose a **DBMS & data model** (e.g., relational).
   - Convert conceptual schema into DBMS schema.

4. **Schema Refinement**
    - Remove redundancy and inconsistencies.
    - Normalize the schema.
5. **Physical Database Design**
    - Improve performance.
    - Add indexes, optimize queries.
    - Decide storage structures.
6. **Security Design**
    - Identify **user groups** and their **roles**.
    - Define access levels (e.g., Bank: Customer – read, Teller – read/update, Manager – full access).

---

7. Database Hosting

- **Where & how database is deployed, stored, and accessed.**
- **Factors to consider**: Scalability, cost, security, performance.
- **Types**:
    - **On-Premise Hosting** – Database hosted locally in organization servers.
    - **Cloud Hosting** – Database hosted online via providers (AWS, Google Cloud, Azure).
- **Cloud Databases**:
    - Advantages – Scalability, remote access, reduced maintenance.
    - Challenges – Security, vendor lock-in, internet dependency.

---

8. Data-Intensive Applications

- **Definition**: Applications that handle **large volumes of data** (storage, retrieval, and processing).
- **Focus**: Quick and consistent access, scalability, reliability.

- **Examples**:
    - Social media platforms (Facebook, Instagram).
    - E-commerce (Amazon, eBay).
    - Banking systems.
    - Real-time analytics platforms.
- **Why DBMS is Critical?**
    - Ensures fast access to large data.
    - Supports multiple concurrent users.
    - Maintains reliability and consistency.
    - Provides scalability for growing data.

Detailed Short Notes – Lecture 02 (IT2140)

SQL – Structured Query Language

---

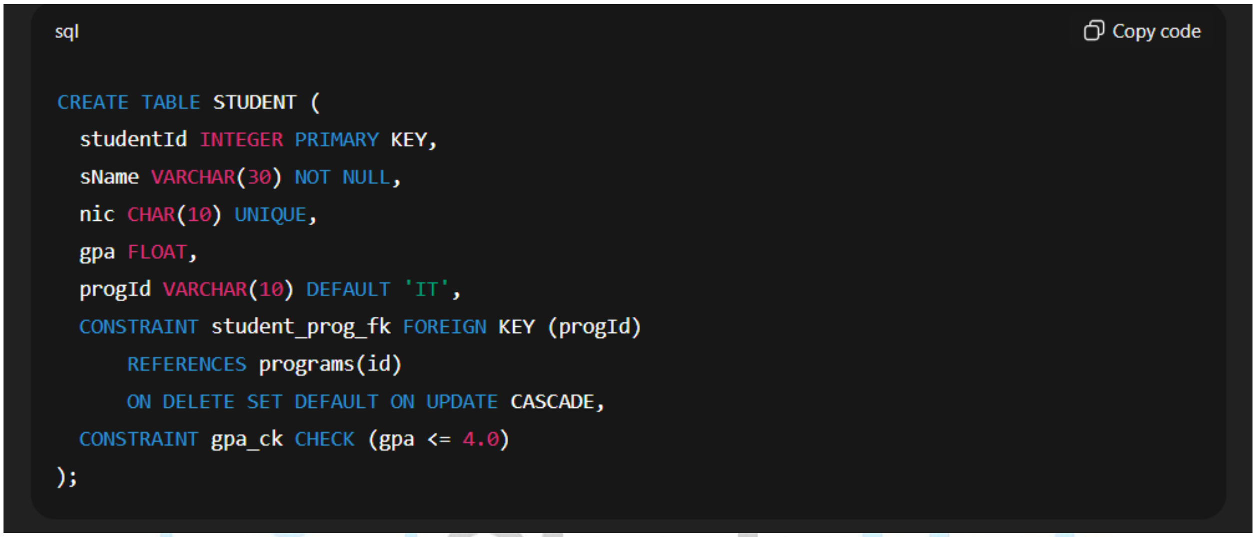
1. Introduction to SQL

- **SQL** = Structured Query Language (originally SEQUEL).
- Developed for **System R** (experimental relational DB system).
- Standardized in **1986** (SQL1/SQL-86). Later revisions: **SQL-92 (SQL2)**.
- **Comprehensive database language** with:
    - **DDL** – Data Definition Language.
    - **DML** – Data Manipulation Language.
    - Security & authorization.
    - Transaction processing.
    - Embedded SQL (integration with programming languages).

2. Data Definition Language (DDL)

- Used to **create, delete, and modify** tables & views.
- Can define **constraints** on tables.

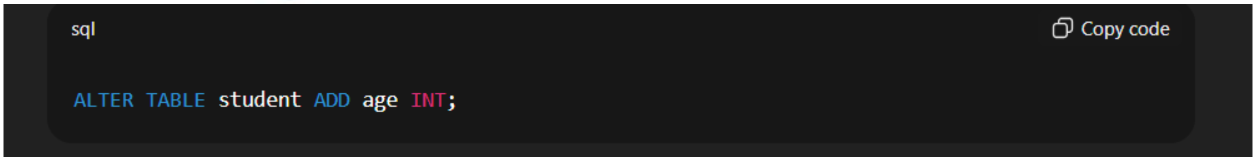
Example – Create Table:



```sql
CREATE TABLE STUDENT (
  studentId INTEGER PRIMARY KEY,
  sName VARCHAR(30) NOT NULL,
  nic CHAR(10) UNIQUE,
  gpa FLOAT,
  progId VARCHAR(10) DEFAULT 'IT',
  CONSTRAINT student_prog_fk FOREIGN KEY (progId)
      REFERENCES programs(id)
      ON DELETE SET DEFAULT ON UPDATE CASCADE,
  CONSTRAINT gpa_ck CHECK (gpa <= 4.0)
);
```

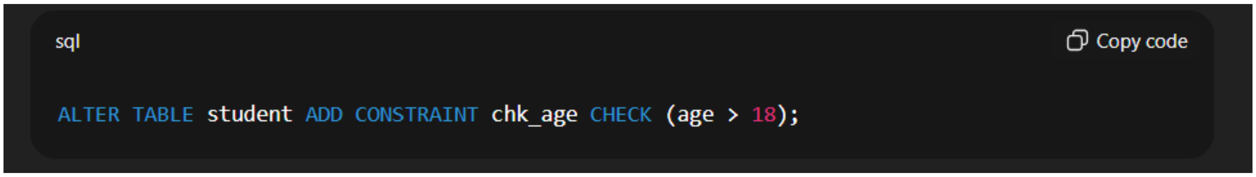
Table Modifications:

- Add new column →



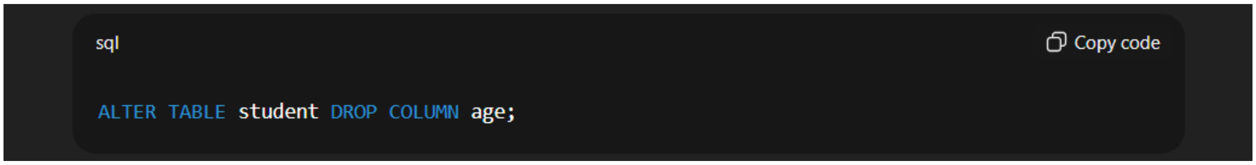
```sql
ALTER TABLE student ADD age INT;
```

- Add new constraint →



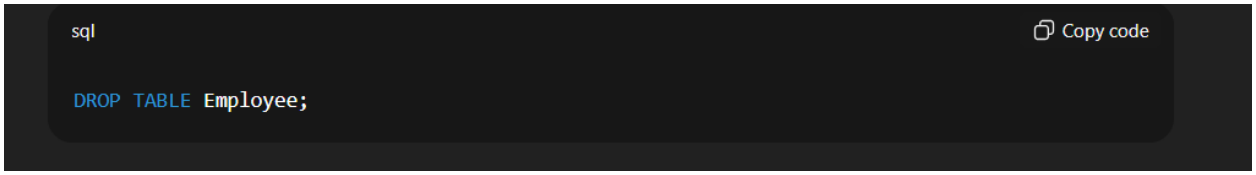
```sql
ALTER TABLE student ADD CONSTRAINT chk_age CHECK (age > 18);
```

- Drop column →



```sql
ALTER TABLE student DROP COLUMN age;
```

- Drop table →



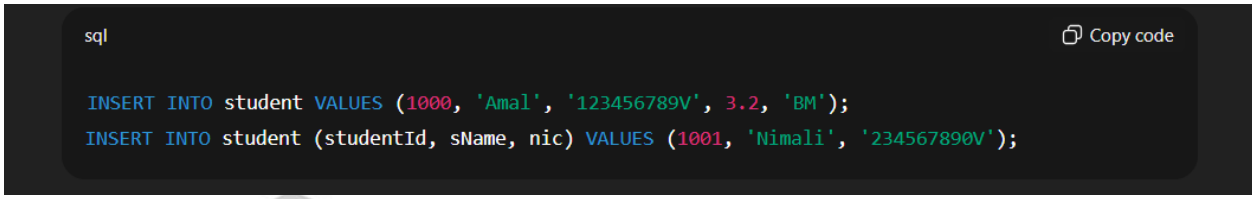
```sql
DROP TABLE Employee;
```

---

3. Data Manipulation Language (DML)

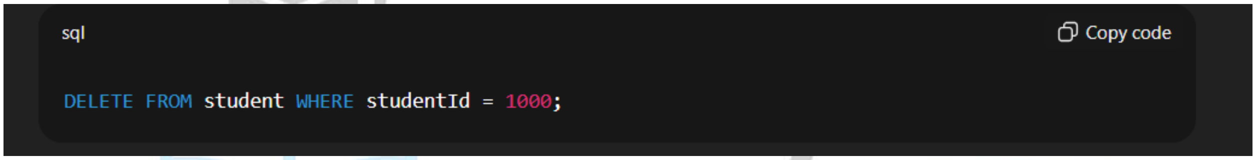
- Allows inserting, deleting, modifying, and retrieving data.

Insert:



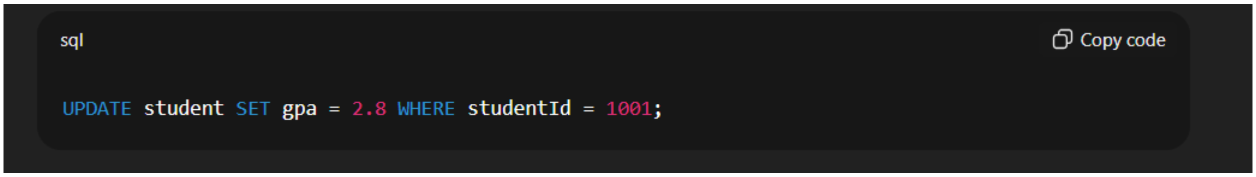
```sql
INSERT INTO student VALUES (1000, 'Amal', '123456789V', 3.2, 'BM');
INSERT INTO student (studentId, sName, nic) VALUES (1001, 'Nimali', '234567890V');
```

Delete:



```sql
DELETE FROM student WHERE studentId = 1000;
```

Update:

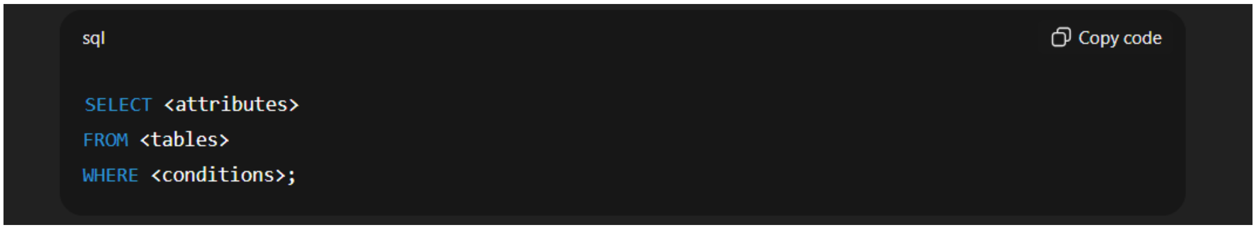


```sql
UPDATE student SET gpa = 2.8 WHERE studentId = 1001;
```

---

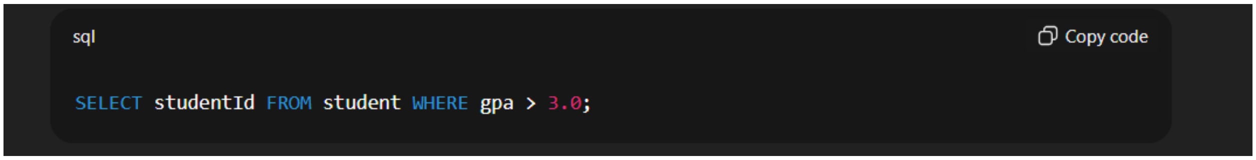
4. SELECT Clause (Querying Data)

Basic form:



```sql
SELECT <attributes>
FROM <tables>
WHERE <conditions>;
```

Example:

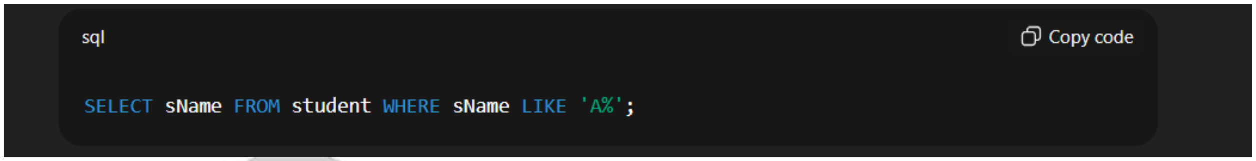


```sql
SELECT studentId FROM student WHERE gpa > 3.0;
```

---

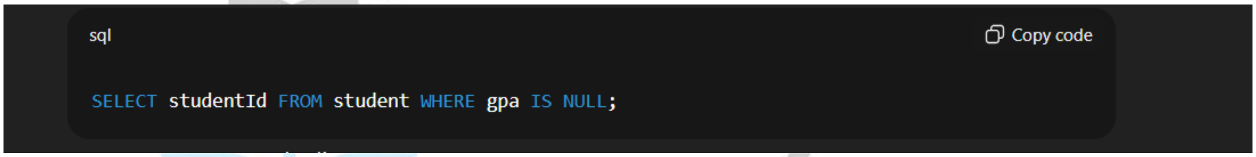
5. Clauses & Operators in SELECT

- **LIKE operator** – Pattern matching.
  - % → any sequence of characters.
  - _ → any single character.



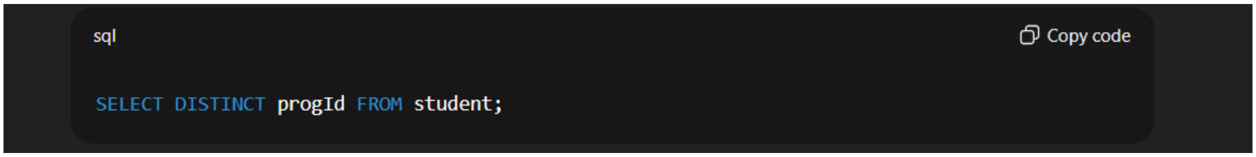
```sql
SELECT sName FROM student WHERE sName LIKE 'A%';
```

- **IS [NOT] NULL** – Check for null values.



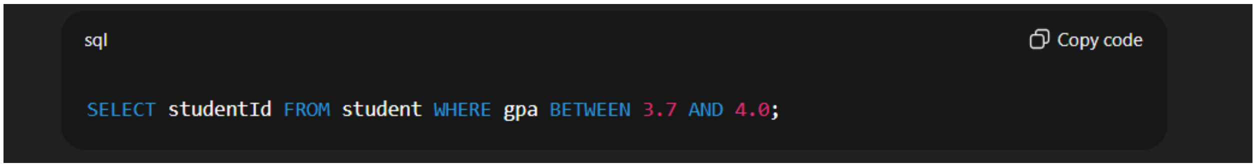
```sql
SELECT studentId FROM student WHERE gpa IS NULL;
```

- **DISTINCT** – Remove duplicates.



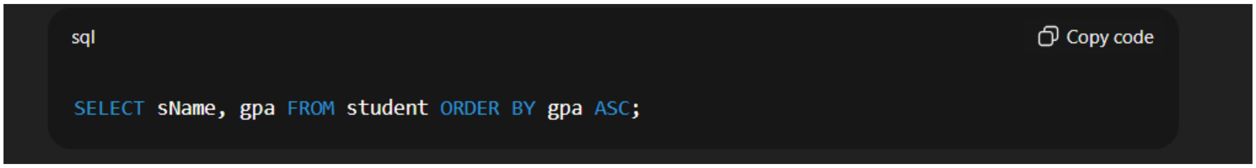
```sql
SELECT DISTINCT progId FROM student;
```

- **BETWEEN** – Range checking.



```sql
SELECT studentId FROM student WHERE gpa BETWEEN 3.7 AND 4.0;
```

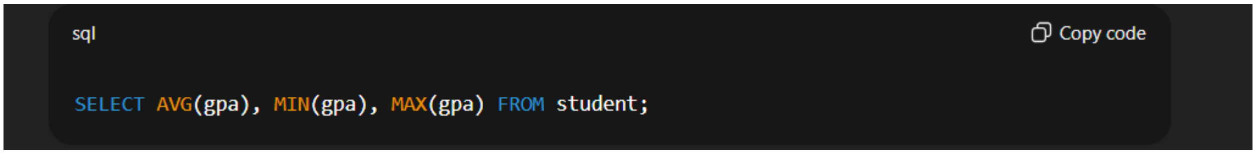
- **ORDER BY** – Sort results.



```sql
SELECT sName, gpa FROM student ORDER BY gpa ASC;
```

6. Aggregation Functions

- **SUM, COUNT, AVG, MIN, MAX**

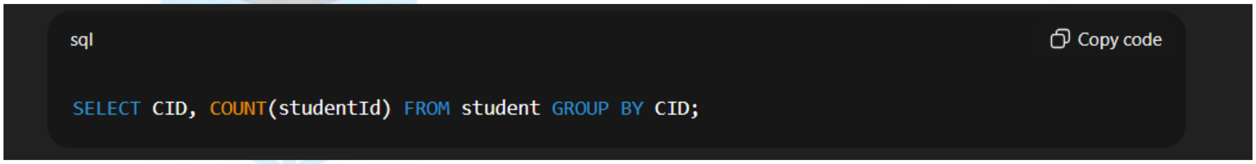


```sql
SELECT AVG(gpa), MIN(gpa), MAX(gpa) FROM student;
```

7. Grouping (GROUP BY)

- Groups rows based on field values.
- Each SELECT item must be:
  - A grouped column.
  - Or an aggregate function.

Example:

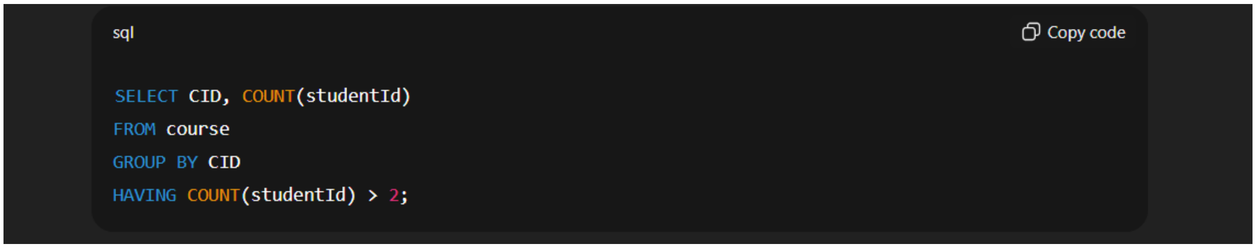


```sql
SELECT CID, COUNT(studentId) FROM student GROUP BY CID;
```

8. HAVING Clause

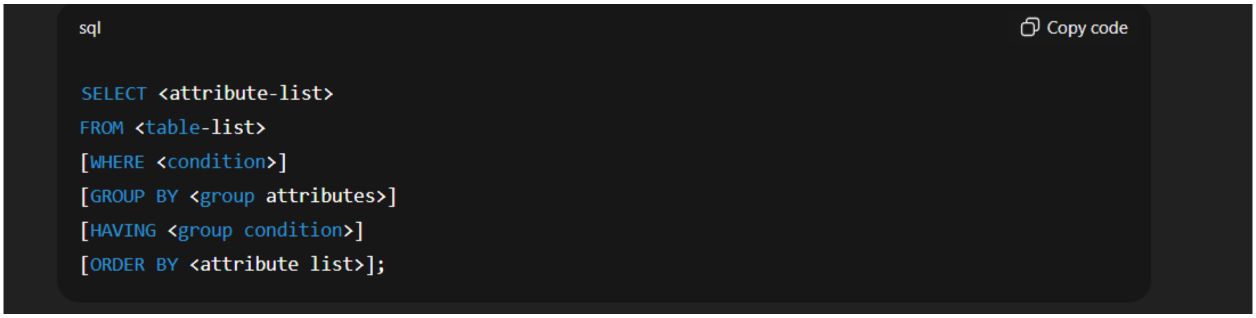
- Applies conditions on grouped data.

Example:



```sql
SELECT CID, COUNT(studentId)
FROM course
GROUP BY CID
HAVING COUNT(studentId) > 2;
```

9. SQL Query Structure (Summary)



```sql
SELECT <attribute-list>
FROM <table-list>
[WHERE <condition>]
[GROUP BY <group attributes>]
[HAVING <group condition>]
[ORDER BY <attribute list>];
```

Detailed Short Notes – Lecture 03 (IT2140)

SQL Joins

---

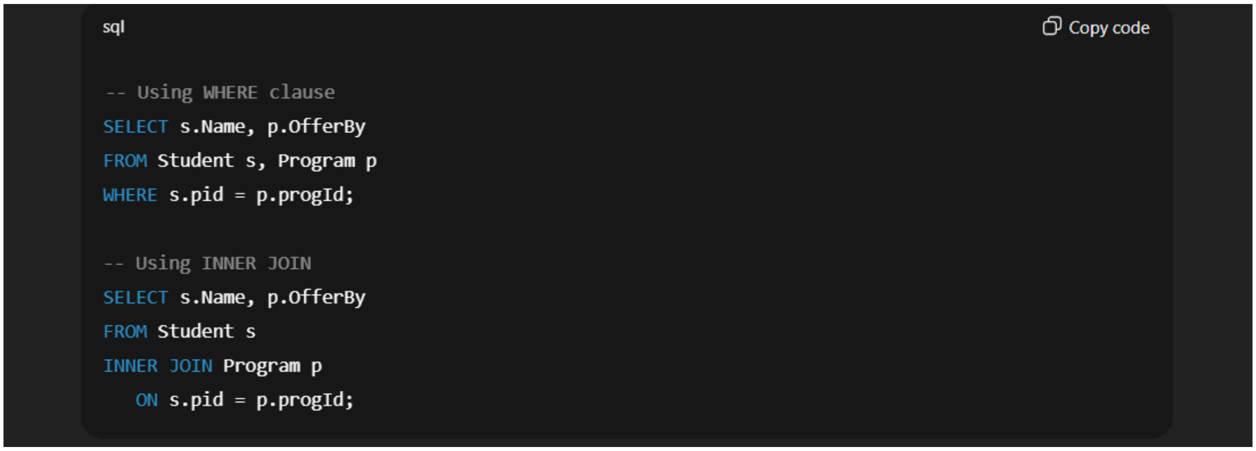
1. Introduction to Joins

- **Purpose**: To **combine data from two or more tables** based on related columns (usually **Primary Key – Foreign Key relationship**).
- Useful when data is spread across multiple tables.

---

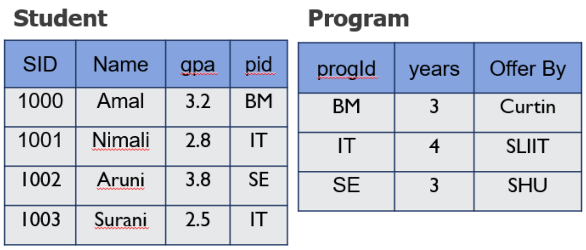
2. INNER JOIN

- Returns only rows with **matching values** in both tables.
- Syntax (two forms):



```sql
-- Using WHERE clause
SELECT s.Name, p.OfferBy
FROM Student s, Program p
WHERE s.pid = p.progId;

-- Using INNER JOIN
SELECT s.Name, p.OfferBy
FROM Student s
INNER JOIN Program p
   ON s.pid = p.progId;
```

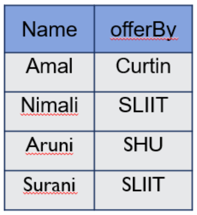


Student

SID	Name	gpa	pid
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	IT

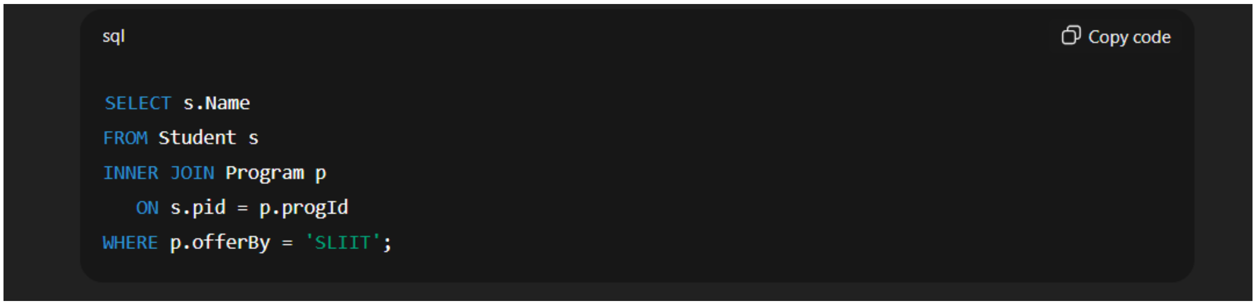
Program

progId	years	Offer By
BM	3	Curtin
IT	4	SLIIT
SE	3	SHU



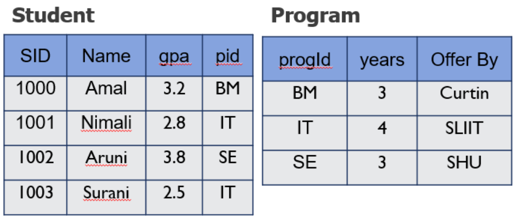
Name	offerBy
Amal	Curtin
Nimali	SLIIT
Aruni	SHU
Surani	SLIIT

- **INNER JOIN with condition** (filtering results):



```sql
SELECT s.Name
FROM Student s
INNER JOIN Program p
   ON s.pid = p.progId
WHERE p.offerBy = 'SLIIT';
```

(Shows names of students in programs offered by SLIIT)

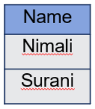


Student

SID	Name	gpa	pid
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	IT

Program

progId	years	Offer By
BM	3	Curtin
IT	4	SLIIT
SE	3	SHU



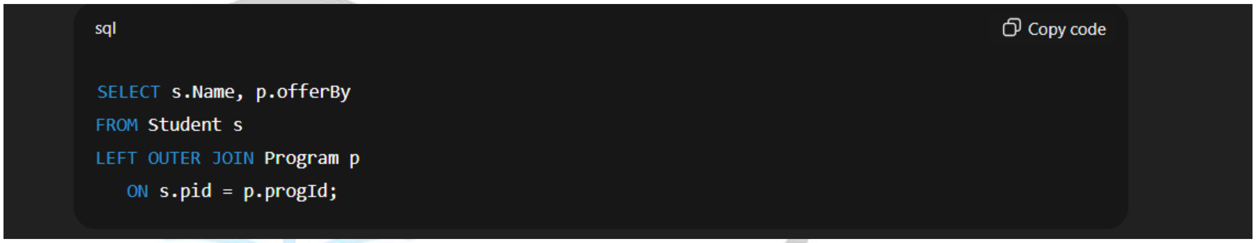
Name
Nimali
Surani

---

3. LEFT OUTER JOIN

- Returns **all rows from the left table**, and matching rows from the right table.
- If no match → result shows **NULL** for right table columns.

Example: Show all students and their offering institute (if available).



```sql
SELECT s.Name, p.offerBy
FROM Student s
LEFT OUTER JOIN Program p
   ON s.pid = p.progId;
```

---

4. RIGHT OUTER JOIN

- Returns **all rows from the right table**, and matching rows from the left table.
- If no match → result shows **NULL** for left table columns.

Example: Show all programs and students enrolled (if any).

```
SELECT s.Name, p.offerBy

FROM Student s

RIGHT OUTER JOIN Program p

  ON s.pid = p.progId;
```

---

5. Summary of Joins

Join Type	What it Returns
INNER JOIN	Only rows with matching values in both tables.
LEFT JOIN	All rows from **left table** + matching rows from right (NULL if no match).
RIGHT JOIN	All rows from **right table** + matching rows from left (NULL if no match).

Detailed Short Notes – Lecture 04 (IT2140)

Conceptual Database Modeling – ER Model

---

1. Conceptual Database Design

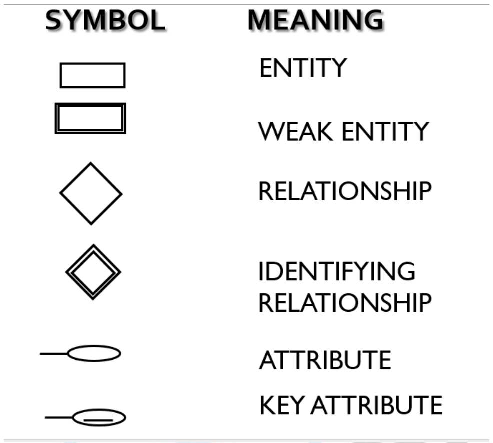
- Comes after **requirement analysis**.
- Creates a **conceptual schema** using a high-level model.
- **Entity–Relationship (ER) model** is the most widely used.
- Output: **ER Diagram** (graphical representation of entities, attributes, and relationships).

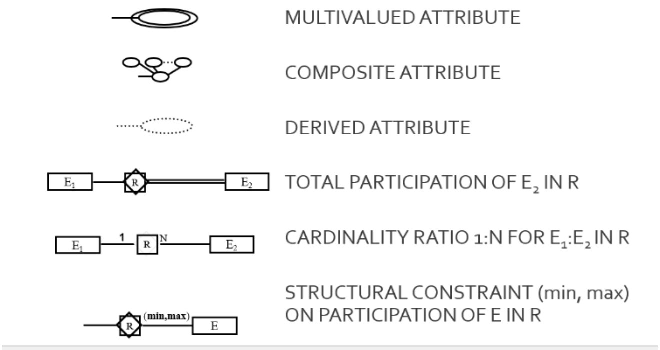
---

2. ER Notation

- **Entity** → Rectangle.
- **Weak Entity** → Double rectangle.
- **Relationship** → Diamond.
- **Identifying Relationship** → Double diamond.
- **Attribute** → Oval.
- **Key Attribute** → Underlined oval.

- **Multivalued Attribute** → Double oval.
- **Composite Attribute** → Oval with sub-attributes.
- **Derived Attribute** → Dashed oval.
- **Participation** → Double line (total participation).
- **Cardinality Ratio** → 1:1, 1:N, N:M.
- **Structural Constraint** → (min, max).




MULTIVALUED ATTRIBUTE
COMPOSITE ATTRIBUTE
DERIVED ATTRIBUTE
E$_1$
R
E$_2$
TOTAL PARTICIPATION OF E$_2$ IN R
E$_1$
1
R
N
E$_2$
CARDINALITY RATIO 1:N FOR E$_1$:E$_2$ IN R
R
(min,max)
E
STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

---

3. Entities & Attributes

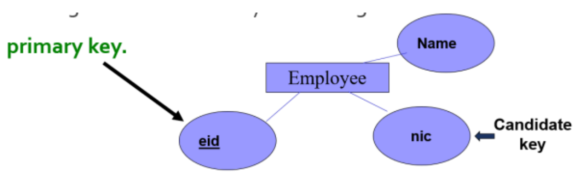
- **Entity** = Object with independent existence (Student, Car, University).
- **Entity Set** = Collection of similar entities.
- **Attributes** = Properties describing an entity.
  - **Domain** – Set of possible values (e.g., age = 0–120).
  - **Composite Attribute** – Divisible (e.g., Name → First, Middle, Last).
  - **Multivalued Attribute** – Multiple values (e.g., multiple emails).
  - **Derived Attribute** – Can be calculated (e.g., Age from DOB).

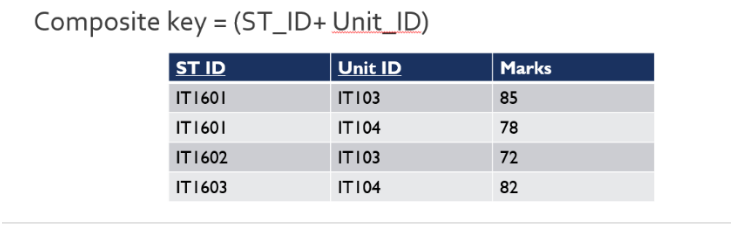
---

4. Keys

- **Key Attribute** – Unique identifier of an entity.

- **Candidate Keys** – Multiple possible unique identifiers.
- **Primary Key** – Chosen candidate key.
- **Composite Key** – Combination of attributes used as a key.
- **Super Key** – Any attribute set that uniquely identifies a tuple (may not be minimal).





Composite key = (ST_ID+ Unit_ID)

ST ID	Unit ID	Marks
IT1601	IT103	85
IT1601	IT104	78
IT1602	IT103	72
IT1603	IT104	82

5. Relationships

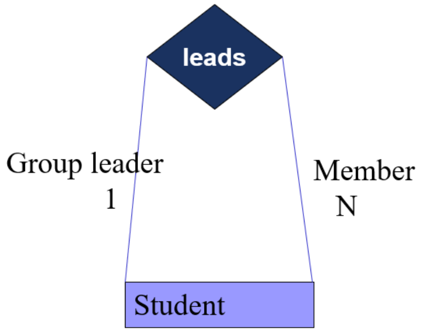
- **Definition:** Association among entities (shown as diamond).
- **Relationship Set** = Collection of relationships of the same type.
- **Degree**:
    - Binary (2 entities).
    - Ternary (3 entities).
    - Quaternary (4 entities).

Cardinality Ratios

- **1:1** → One employee manages one department.
- **1:N** → One department has many employees.
- **N:M** → Many employees work on many projects.

Recursive Relationship

- Relationship among entities of the same set.
    - Example: *Student leads Student* (group assignments).
    - Example: *Employee supervises Employee*.



Descriptive Attributes

- Attributes that describe the relationship.
- Example: Employee **works in Department since 2008** (attribute = "since").

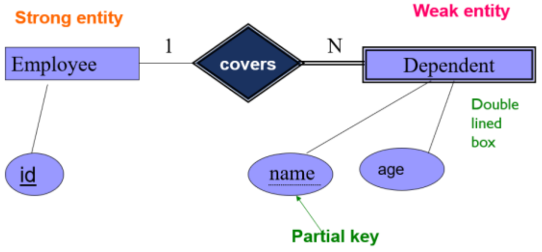
---

6. Participation Constraints

- **Total Participation** – Entity must participate (double line).
  - Example: Student must register in a degree.
- **Partial Participation** – Entity may or may not participate.
  - Example: Department may or may not have employees.

---

7. Weak Entities



- Cannot exist independently.
- Depend on another entity (owner entity).
- Characteristics:
  - No key attributes → identified by **partial key** + owner's key.

  - Connected via **identifying relationship**.
  - Always has **total participation**.

---

8. Enhanced ER (EER) Model

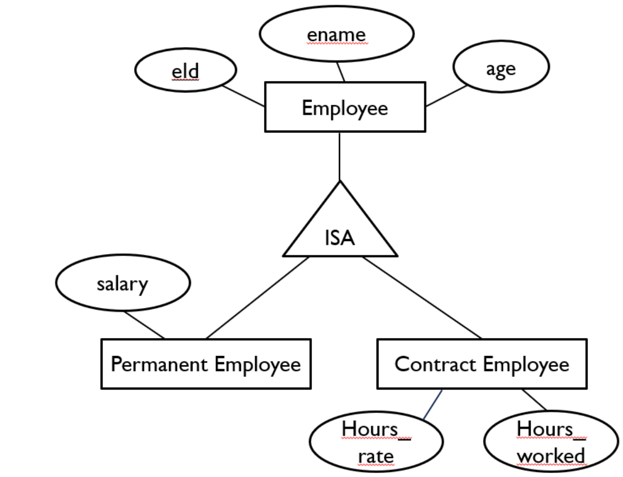
Specialization

- Defining sub-classes of an entity.
- Example: Employee → Permanent Employee, Contract Employee.

Generalization

- Grouping similar entities into a super-class.
- Example: Student + Faculty → Person.

ISA Relationships



- Subclasses inherit attributes of superclass.
- Example: Permanent Employee ISA Employee.

Constraints in Specialization

- **Overlapping Constraint**: Can entity belong to multiple subclasses?
  - Example: Person can be both *Student* and *Faculty* (overlapping).
- **Covering Constraint**: Do subclasses cover all superclass entities?

    - **Total Specialization** – Every entity belongs to a subclass.
    - **Partial Specialization** – Some entities may not belong to any subclass.

---

9. Summary of Key Points

- ER model → Foundation of database design.
- Entities = Objects, Attributes = Properties, Relationships = Associations.
- Keys identify entities uniquely.
- Participation and cardinality define relationship rules.
- Weak entities depend on strong entities.
- EER adds specialization/generalization for advanced modeling.

🚀 Mock Exam for Database Design and Development Now Live on StudyHub!

👉 Take the Exam Here: https://edulk.site/

📢 Stay Updated with StudyHub

For continuous updates on mock exams and resources, join our WhatsApp group here:

👉 Join WhatsApp Group - https://chat.whatsapp.com/D8mkgq8B9rh0Fpw3mw8Odk?mode=ems_copy_c

📌 What's Inside:

💻 Realistic Exam Format — same structure as your mid-term paper

📝 Multiple Choice Questions (MCQs) — test your knowledge

📖 Detailed Solutions — review every answer to strengthen understanding

Good luck & happy studying! 🍀🎓

⚠ Disclaimer: This mock exam is independently provided and not affiliated with any official university examinations.