FRANKLIN'S LECTURES

DATABASE MANAGEMENT SYSTEM

Module 2

CST 204

Relational Model

- Structure of Relational Databases
- Integrity Constraints
- Synthesizing ER Diagrams to Relational Schemas
- Relational Algebra:
  - Select, Project, Cartesian Product
  - Equi-Join, Natural Join
  - Query Examples
- SQL Basics:
  - Introduction to SQL
  - DDL commands (CREATE, DROP, ALTER)
  - Basic INSERT, DELETE, UPDATE

Very Important Topics (Frequently Asked)

Topic	Related Questions from Past QPs
Relational Algebra Queries	2021 Q13a, 2022 Q13a, 2023 Q13a - Always present with varied schemas like STUDENT/EMPLOYEE/SAILOR etc.
SQL DDL - Table Creation, Constraints	2022 Q14a, 2024 Q14a - Asked to define schema with constraints and perform SQL operations. 2022 Q14b, 2024 Q14a(ii) - Common across all years.
Integrity Constraints (Primary, Foreign, Not Null)	2022 Q14b, 2024 Q14a(ii) - Common across all years.
Equi-Join vs Natural Join	2022 Q4 (Part A), 2024 Q4

Moderately Important Topics

Topic	Related Questions from Past QPs
Converting ER to Relational Schema	2022 Q13, 2024 Q13 - Sometimes merged with relational algebra or SQL.
SQL INSERT, UPDATE, DELETE, ALTER 2022 Q14a(ii-iv) usage	2022 Q14a(ii-iv)
SQL Query Examples (basic)	2022 Q14b, 2023 Q13b - Basic INSERT/UPDATE/DELETE uses.
Foreign Key Importance	2023 Q13b, 2024 Q14b - Not always separately asked, but often embedded in schema Qs.

Less Important Topics

Topic	Related Questions
Cartesian Product in Relational Algebra (explicit Qs rare)	2Rarely asked directly in QPs
Definitions of Relational DB Concepts (structure only, no examples)	Sometimes appears in Part A (e.g., 2022 Q3, 2023 Q2)

Summary

Topic	Topic
Very Important	Relational Algebra Queries, SQL Table Definitions, Integrity Constraints, Equi-Join vs Natural Join
Moderate	ER Relational Conversion, SQL Data Operations (INSERT, UPDATE, DELETE), Foreign Key Explanation
Less important	Cartesian Product, Basic Relational DB Definitions

What is Relational Algebra?

Definition:
A formal language for manipulating relations in a database.

Core Operations:

- SELECT (σ),
- PROJECT (π),
- JOIN ($\bowtie$) ,
- CARTESIAN PRODUCT ($\times$)

What is Relational Algebra?

1. Select (σ) - Filters rows based on a condition.
2. Project (π) - Selects specific columns from a relation.
3. Cartesian Product (×) - Combines every tuple of one relation with every tuple of another.
4. Union (U) - Combines tuples from two relations with duplicate elimination.
5. Set Difference (-) - Finds tuples in one relation but not in another.
6. Rename (ρ) - Renames a relation or its attributes.

FRANKLIN'S LECTURES

SQL

```
select * from student
where
age > 21
```

$\pi_{name}(\sigma_{age>21}(STUDENT))$

student

id	name	age	marks
1	A	25	50
2	B	20	45

$\pi_{name, age}(STUDENT)$

INDIA'S NO 1 ENGINEERING EDUCATION PLATFORM
FRANKLIN'S LECTURES

Schema: `Sailors(sid, sname, rating, age)`

$\sigma_{rating \geq 8}$ (Sailors) $\longrightarrow$ Select sailors with rating ≥ 8
π_{sname} (Sailors) $\longrightarrow$ Project sailor names only

Exercise:

"Find names of sailors who reserved boat 101"

SQL DDL – Table Definition Basics

DDL Commands:

- CREATE: Define structure
- DROP: Remove table
- ALTER: Modify table

```
Example :
CREATE TABLE Student (
  RollNo INT PRIMARY KEY,
  Name VARCHAR(50),
  Age INT
);
```

"Which DDL command will you use to add a column?"

Integrity Constraints

Important Constraints:

- PRIMARY KEY
- FOREIGN KEY
- UNIQUE
- NOT NULL
- CHECK

Example:

```
FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
```

Activity:

"What happens if we try to delete a referenced department?"

FRANKLIN'S LECTURES

NOT NOW

Student

id	name	universityregNo	adhar
		FCB1	1234 1234

1
2
3
4
5
6

PK

Union

UDC

PK

UC

UC

PK

INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM

Joins – Equi-Join vs Natural Join

Type	Description	Duplicate Colums?
Equi-Join	Join using = condition	✅ Yes
Natural Join	Auto-join on columns with same name	❌ No

EXAMPLE:

```
SELECT * FROM Student JOIN Department ON Student.DeptID
= Department.ID;
```

FRANKLIN'S LECTURES

Equi Join

student

id	name	depid
1	A	1
2	B	2

department

id	name	location
1	CSE	Calicut
2	ME	Kannur

```
select * from student
        join department
on
student.deptid = department.id
```

INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM
FRANKLIN'S LECTURES

Natural Join

student

id	name	deptid

department

deptid	deptname	deptlocation

select * from student
join department

FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM

Past Paper Highlights

Year	Question Focus
2021	RA + SQL table creation
2022	Joins + Relational Algebra
2023	Primary, Foreign Keys + Algebra
2024	Join types + Constraints

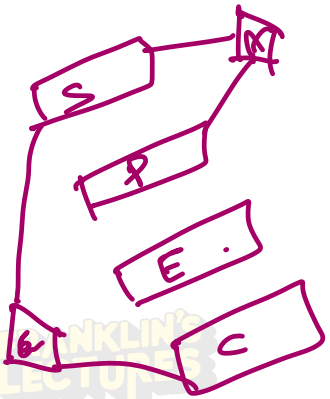
Summary

Mini Quiz:

- Write a relational algebra query to fetch student names older than 20
- What is the use of ALTER command?
- Give a difference between Equi-Join and Natural Join

Given Schema:

STUDENT(ROLLNO, NAME, AGE, GENDER, ADDRESS, ADVISOR)
COURSE(COURSEID, CNAME, CREDITS)
PROFESSOR(PROFID, PNAME, PHONE)
ENROLLMENT(ROLLNO, COURSEID, GRADE)



- ADVISOR in STUDENT is a foreign key referring to PROFESSOR(PROFID).
- ROLLNO and COURSEID in ENROLLMENT are foreign keys referencing STUDENT and COURSE, respectively.

1. Retrieve the names of all female students.

✅ **Answer:**

To get the names of all students whose gender is female (`GENDER = 'F'`), we apply a selection followed by a projection:

$$\pi_{NAME}(\sigma_{GENDER = 'F'}(STUDENT))$$

$\sigma_{GENDER = 'F'}(STUDENT)$: Selects all tuples from STUDENT where gender is female.
$\pi_{NAME}(\ldots)$: Projects only the NAME attribute from the result.

$$\Pi_{name}\left(\sigma_{gender='F'}\left(STUDENT\right)\right)$$

FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM

2. Retrieve the names of all male students along with the name of their advisor.

✅ **Answer:**

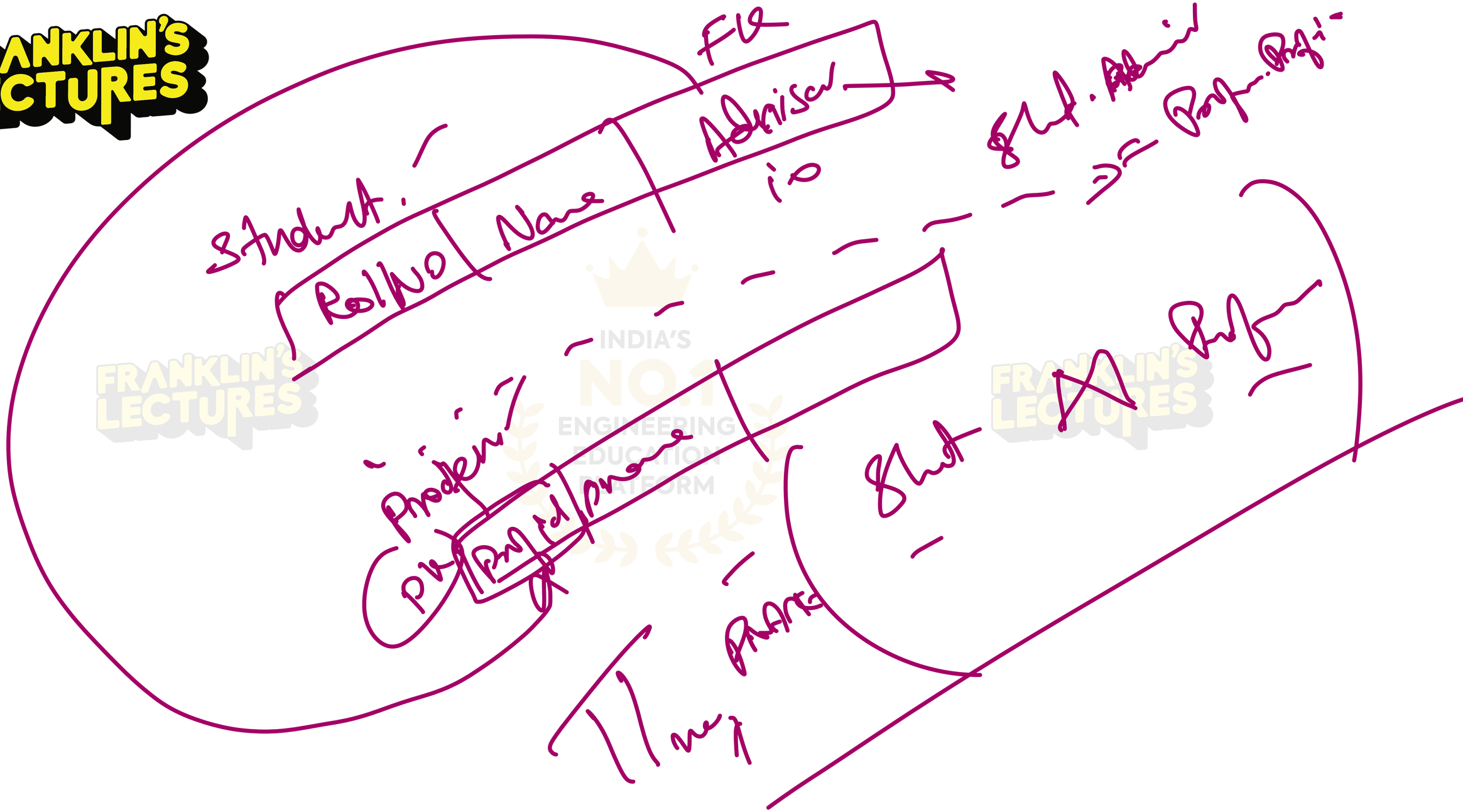
To get male students along with their advisor's name, we:

1. Filter STUDENT relation for GENDER = 'M'.
2. Perform a natural join (or theta join) with PROFESSOR on STUDENT.ADVISOR = PROFESSOR.PROFID.
3. Project the student name and professor (advisor) name.

$\pi_{NAME, PNAME}(\sigma_{GENDER = 'M'}(STUDENT) \bowtie STUDENT.ADVISOR = PROFESSOR.PROFID)$

```
σ GENDER = 'M'(STUDENT): Selects male students.
  STUDENT.ADVISOR = PROFESSOR.PROFID: Joins with PROFESSOR to get advisor details.
π NAME, PNAME(...): Projects student name and advisor (professor) name.
```

FRANKLIN'S LECTURES



FRANKLIN'S LECTURES

INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM

3. Retrieve the roll numbers and names of students who have not enrolled for any course.

To find students who are not enrolled in any course, we:

- Take all students from STUDENT.
- Subtract the students who appear in the ENROLLMENT relation.
- Use set difference operator (–).

R_1 R_2

π_ROLLNO, NAME(STUDENT) − π_ROLLNO, NAME(STUDENT ENROLLMENT)

Left side: All students' roll numbers and names.

Right side: Students who have enrolled (using join with ENROLLMENT).

Set difference gives us students not enrolled in any course.

$R_1 = \{1, 2, 3\}$

$R_2 = \{1, 2\}$

$R_1 - R_2 \Rightarrow 3$

R_1 = all students

R_2 = enrolled students

$R_1 - R_2$

$\sigma_{gender = 'F'}(STUDENT)$

$\pi_{name}(STUDENT \bowtie PROFESSOR)$

```
SELECT NAME
FROM STUDENT
WHERE GENDER = 'F';
```

π_NAME(σ_GENDER = 'F'(STUDENT))

```
π_NAME, PNAME(σ_GENDER = 'M'(STUDENT)
  ⋈ STUDENT.ADVISOR = PROFESSOR.PROFID)
```

```
SELECT S.NAME, P.PNAME
FROM STUDENT S
JOIN PROFESSOR P ON S.ADVISOR = P.PROFID
WHERE S.GENDER = 'M';
```

```
π_ROLLNO, NAME(STUDENT)
- π_ROLLNO,⋈
NAME(STUDENT ENROLLMENT)
```

$$\pi_{ROLLNO,\,NAME}(STUDENT) - \pi_{ROLLNO,\,NAME}(STUDENT \bowtie_{ENROLLMENT})$$

```
SELECT ROLLNO, NAME
FROM STUDENT
WHERE ROLLNO NOT IN (
   SELECT DISTINCT ROLLNO
   FROM ENROLLMENT
);
```

- Sailors(sid, sname, rating, age)
- Boats(bid, name, color)
- Reserves(sid, bid, day)

- Each Sailor may reserve multiple Boats
- Each reservation has a date
- sid and bid are foreign keys in the Reserves table

Write relational algebra queries(UQ-Solution):

i) Find the name of sailors who have rating at least

$\pi_{SNAME}(\sigma_{RATING} \geq 7\ (SAILORS))$

$\Pi_{SNAME}(\sigma_{RATING \geq 7}(SAILORS))$

ii) Find the name of sailors who have reserved the boat with ID 110

$\pi_{SNAME}((SAILORS \bowtie RESERVES) \bowtie \sigma_{BID = 110}(BOATS))$

Name

110

1. Retrieve the name and address of all employees who work for the 'Research' department.

$$\text{RESEARCH_DEPT} \leftarrow \sigma_{\text{Dname='Research'}}(\text{DEPARTMENT})$$

$$\text{RESEARCH_EMPS} \leftarrow (\text{RESEARCH_DEPT} \bowtie_{\text{Dnumber=Dno}} \text{EMPLOYEE})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Address}}(\text{RESEARCH_EMPS})$$

As a single in-line expression, this query becomes:

$$\pi_{\text{Fname, Lname, Address}}(\sigma_{\text{Dname='Research'}}(\text{DEPARTMENT} \bowtie_{\text{Dnumber=Dno}} (\text{EMPLOYEE}))$$

Converting ER Model to Relational Schema

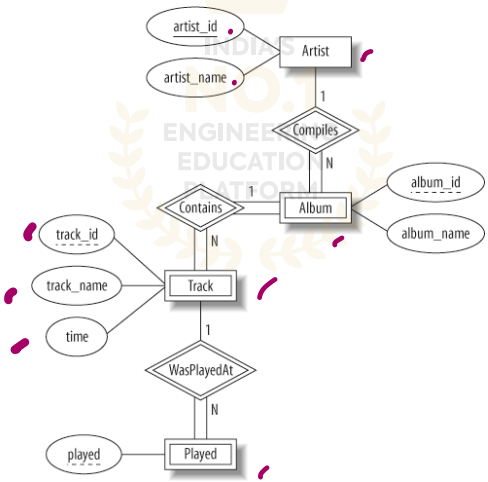
<u>Steps to Convert:</u>

1. Entities → Tables
2. Attributes → Columns
3. Primary Key → Underline in ERD → PRIMARY KEY
4. Relationships → Foreign Keys / New tables (for M:N)

<u>**Example Activity:**</u>

Convert a Student–Enrolled–Course ERD to tables.

Converting ER Model to Relational Schema



SQL – Data Manipulation Commands

Command	Purpose	Example
INSERT	Add new data	INSERT INTO Student VALUES (...);
DELETE	Remove data	DELETE FROM Student WHERE ID=1;
UPDATE	Modify data	UPDATE Student SET Age=21 WHERE ID=2;

"What will happen if you delete a student referenced in another table?"

SQL - INSERT

```
CREATE TABLE Students (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    age INT,
    course VARCHAR(50)
);
```

INSERT – To add new records into the table

```
INSERT INTO Students (id, name, age, course)
VALUES (1, 'Raul', 21, 'Computer Science');
```

You can insert multiple rows like this:

```
INSERT INTO Students (id, name, age, course)
VALUES
(2, 'Aisha', 22, 'Electronics'),
(3, 'John', 20, 'Mechanical');
```

SQL – Data Manipulation Commands

```
UPDATE Students
SET age = 23, course = 'Data Science'
WHERE id = 1;
```

This updates Rahul's age and course where `id = 1`.

DELETE – To remove records

```
DELETE FROM Students
WHERE id = 2;
```

This deletes the record of the student with id = 2.

DDL

Add a new column:

```
ALTER TABLE Students
ADD email VARCHAR(100);
```

Change a column's data type:

```
ALTER TABLE Students
MODIFY age SMALLINT;
```

Rename a column (MySQL):

```
ALTER TABLE Students
CHANGE name full_name
VARCHAR(100);
```

Drop a column:

```
ALTER TABLE Students
DROP COLUMN email;
```

DROP – To delete an entire table, column, or database

```
DROP TABLE Students;
```

Drop a column from a table

```
ALTER TABLE Students
DROP COLUMN email;
```

FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM

Simple SQL SELECT Queries

SELECT Syntax:

```
SELECT name FROM Student WHERE age > 20;
```

```
Use WHERE clause to filter
Use AND/OR for multiple conditions

Teaching Tip: Ask students to modify queries and predict results
```

1.Select all students aged 22:

```
SELECT * FROM Students
WHERE age = 22;
```

2. Select students from "Computer Science":

```
SELECT name, course FROM
Students
WHERE course = 'Computer
Science';
```

3.Select students where age is not 22:

```
SELECT * FROM Students
WHERE age <> 22;
```

4. Using AND / OR in WHERE:

```
SELECT * FROM Students
WHERE course = 'Computer
Science' AND age = 22;
```

2. What is a Foreign Key in DBMS?

A **foreign key** is a column (or set of columns) in one table that **refers to the primary key** in another table.

It is used to **link two tables** and maintain **referential integrity** between them.

Foreign Key (Example)

PK

student		
student_id	name	age
1	Rahul	21
2	Aisha	22
3	john	20

FK

Enrollments		
enrollments_id	student_id	course
1	1	Math
2	1	Physics
3	2	Chemistry

In this case::

- Students.student_id is the **Primary Key** in Students.
- Enrollments.student_id is a **Foreign Key** that refers to Students.student_id.

SQL Syntax to Define Foreign Key

```
CREATE TABLE Enrollments (
        enrollment_id INT PRIMARY KEY,
        student_id INT,
        course VARCHAR(100),
        FOREIGN KEY (student_id) REFERENCES
Students(student_id)
);
```

- You **can't insert** a student_id in Enrollments that **does not exist** in Students.
- If you **delete a student** from Students, you must **handle** the enrollments related to that student (cascade or restrict).

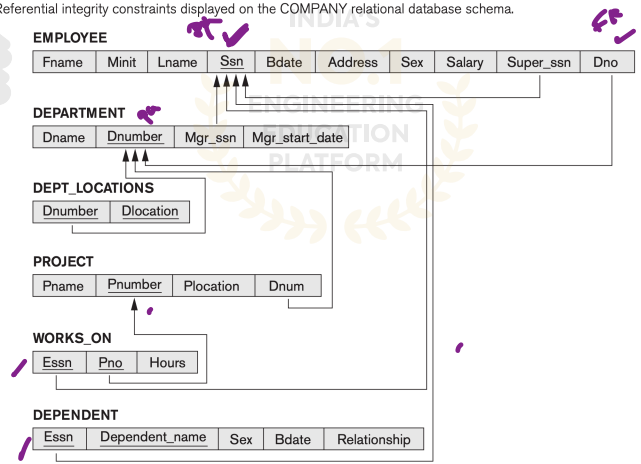
Understanding FOREIGN KEYS

Definition:
A field that creates a relationship between two tables
→ Links child to parent

FOREIGN KEY (DeptID) REFERENCES
Department(DeptID)

Relational Schema with Foreign Key Constraint

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Topic	Related Questions
Cartesian Product in Relational Algebra (explicit Qs rare)	Rarely asked directly in QPs
Definitions of Relational DB Concepts (structure only, no examples)	Sometimes appears in Part A (e.g., **2022 Q3**, **2023 Q2**)

Cartesian Product in Relational Algebra

Definition:

A **Cartesian Product** is a combination of all rows from two relations. The result will have all possible combinations of rows from the tables.

Example:

```
SELECT * FROM Student, Course;
```

What is Cartesian Product in Relational Algebra?

The **Cartesian Product** (also called **Cross Product**) is a binary operation denoted by **×**.

It combines **every tuple (row)** in one relation with **every tuple** in another relation.

Notation: R × S

R and S are two relations (tables).

The result is a new relation that contains **all combinations of tuples** from R and S.

What is Cartesian Product in Relational Algebra?

Relation R (Students)	
sid	**name**
1	Rahul
2	Aisha

Relation S (Courses)	
cid	**course**
101	Math
102	Physics

R × S Result (Cartesian Product)			
sid	**name**	**cid**	**course**
1	Rahul	101	Math
1	Rahul	102	Physics
2	Aisha	101	Math
2	Aisha	102	Physics

✅ Each row from R is combined with **every row** from S

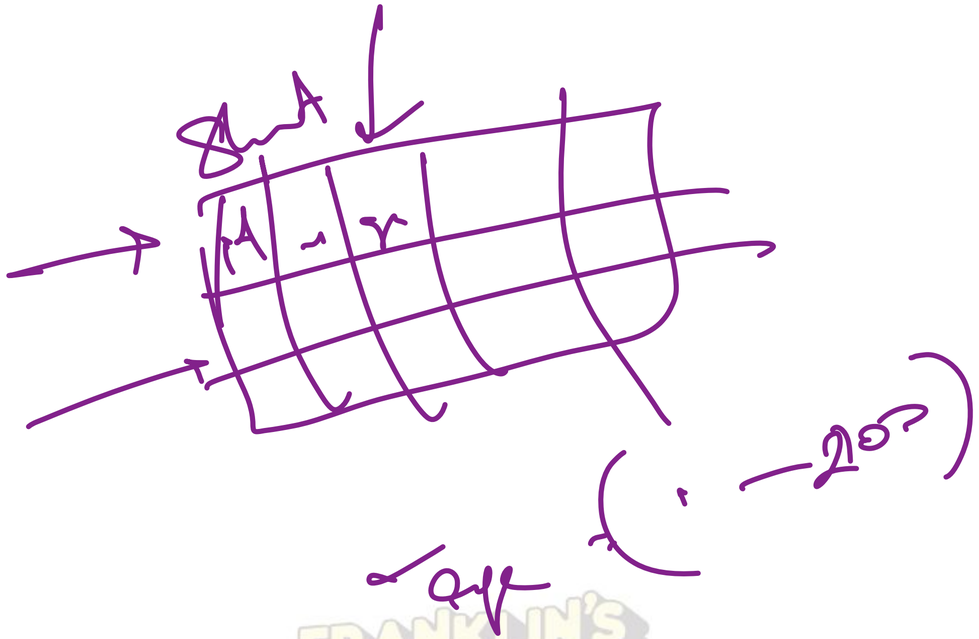
Characteristics:

- **Number of Rows =** `rows_in_R × rows_in_S`
- **Number of Columns =** `columns_in_R + columns_in_S`
- **Often used as an intermediate step in JOIN operations (especially in Theta Join, Equi Join, Natural Join)**

⚠️ Use Case:

We rarely use Cartesian Product alone.
Instead, we combine it with **selection (σ)** to get **meaningful joins**.

Relational Database Definitions



Key Concepts (Briefly):

- **Relation** – Table with rows and columns
- **Tuple** – Row in a table
- **Attribute** – Column in a table
- **Domain** – Set of possible values for an attribute

"How is a table in relational database different from a regular table in Excel?"

SQL Syntax Variations – ALTER TABLE

Commands Overview:

- **ADD** – Add a new column
  DROP – Remove a column
  MODIFY – Change the datatype of a column

Example:

"How would you change a column from INT to VARCHAR?"

Why These Topics Are Less Important

Cartesian Product: Rarely tested directly. Used in explaining joins and query optimization.

Relational Definitions: Touched upon in theory but not often in exam questions.

SQL Modifications (ALTER variations): Rarely asked as major questions; usually part of a larger question.

"When would you use **ALTER TABLE** to modify a column?"

Past Paper Insights

Topic	QPs
Cartesian Product	Rarely tested directly in 14-mark Qs
SQL Syntax Details (ALTER)	Sometimes used in sub-questions or part A
Basic Relational DB Definitions	Touched upon briefly in the beginning

Summary & Revision Tips

Keep a high-level understanding of definitions and SQL syntax variations.

Practice Cartesian Product as part of larger queries, even if it's rarely asked directly.

Focus on relational algebra basics and how they apply to SQL queries.

Topics Covered

Syllabus Topic	QP Coverage	Should You Teach It?
Set operations in Relational Algebra (UNION, INTERSECT, MINUS)	Rarely asked directly	✔️ Yes — core RA concept
Cartesian Product in Relational Algebra	Only mentioned indirectly	✔️ Yes — part of Join logic
Use of DISTINCT, ORDER BY in SQL	Practically used, not examined	✔️ Good for real-world SQL
NULL value handling in SQL (IS NULL, IS NOT NULL)	Not asked	⚠️ Optional, but good to mention
Constraints like CHECK	Rarely emphasized	⚠️ Mention briefly in examples
ER to Relational Conversion – handling weak entities	Not isolatedly asked	✔️ Yes — links Module 1 & 2

FRANKLIN'S LECTURES

Thank you