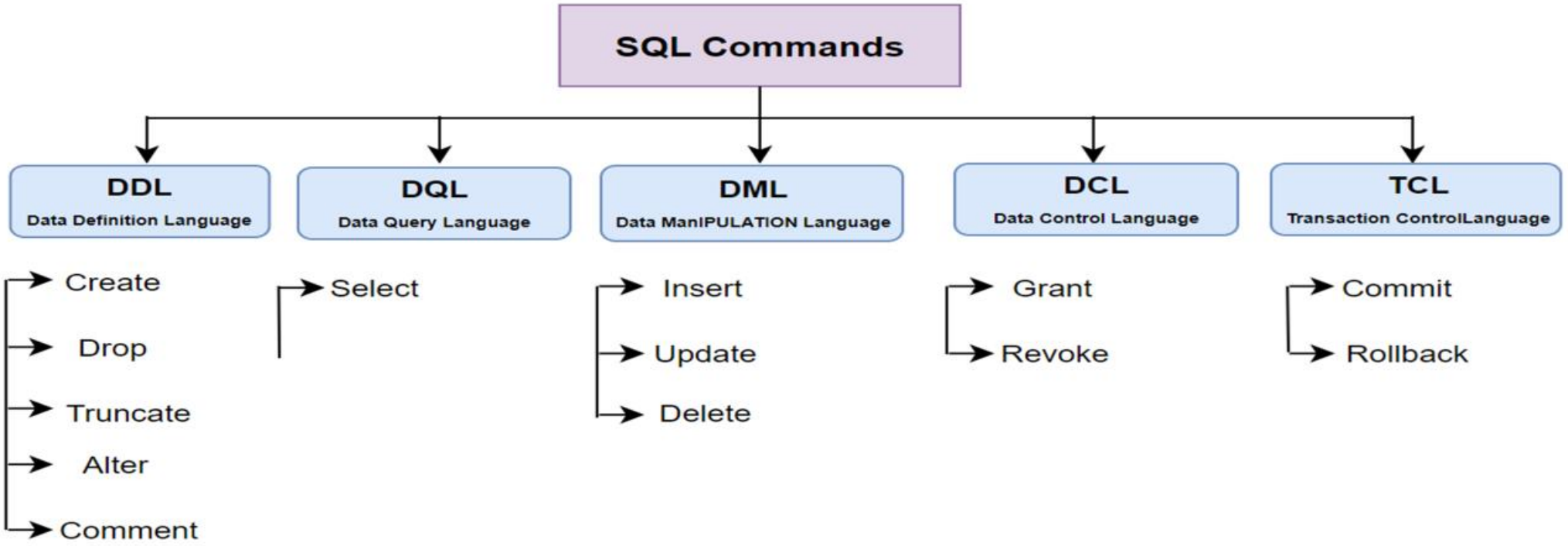
Winter Winning Camp

DATABASE MANAGEMENT SYSTEM

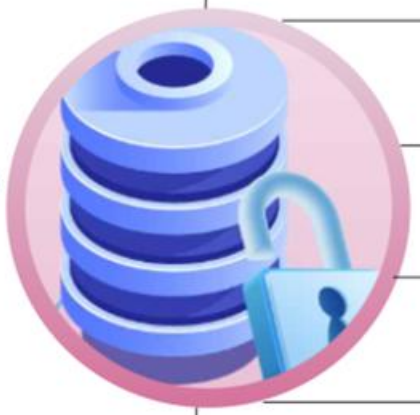
Department Of Career Planning & Development

SQL – Structured Query Language

- SQL is a language just like C++ and Java. It is actually used for accessing and manipulating the databases.
- We have different data definition commands in SQL for accessing and modifying the databases. This includes: DDL, DQL, DML, DCL and TCL.



Constraints in SQL



Primary Key Constraint

```
Create Table Students (
StudentID INT Primary Key,
Name VARCHAR(20),
Age INT );
```

Foreign Key Constraint

Unique Constraint

```
Create Table Employee (
EmpID INT Primary Key,
Email VARCHAR(20) UNIQUE,
Name VARCHAR(20) );
```

Check Constraint

```
Alter Table Students
ADD CONSTRAINT CheckAge CHECK (Age>=18);
```

Default Constraint

```
Create Table Employee (
EmpID INT Primary Key,
Email VARCHAR(20) UNIQUE,
Name VARCHAR(20)
Department VARCHAR(30) DEFAULT 'Production');
```

Not Null Constraint

```
Create Table Customers (
ID INT Primary Key,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL );
```

Contents

Note: Problem statements & tasks are included after each topic.

Views

1. VIEW IS A DATABASE OBJECT
2. VIEW IS CREATED OVER AN SQL QUERY
3. VIEW DOES NOT STORE ANY DATA
4. VIEW IS LIKE A VIRTUAL TABLE
5. DATA ABSTRACTION->Data abstraction in views means

hiding the complexity of database tables and presenting only the required information to the user, using a view.

6. SECURITY : ROW LEVEL & COLUMN LEVEL SECURITY

Row level security: restricts **rows** (with WHERE clause).

Column level security: restricts **columns** (by selecting only some columns).

- So, a **view is a security layer** where you decide:
- Which **rows** a user can access (row-level).
- Which **columns** a user can access (column-level).

Syntax of Views Postgre:

```
CREATE VIEW student_view AS
SELECT studentid, studentname, city
FROM student;
```

MATERIALIZED VIEW:

(1) Takes disk space

(2) A **materialized view (MV)** is different: it **stores the actual result of the query physically**

(3) Since results are precomputed and stored, queries on a materialized view run much faster.(Performance Boost)

(4) Materialized views store physical data and cannot be **updated and Inserted Data directly.**

- **Refresh needed** – Data in a materialized view can become stale(Out dated or not Fresh) if base tables change. That's why we use:
- REFRESH COMPLETE (rebuilds the whole MV)

Normal View	Materialized View
Stores only query	Stores actual data
Always up-to-date	Can become stale
No storage needed	Needs storage space
Slower (query runs each time)	Faster (precomputed results)

Syntax:

```
--Materialized View
CREATE MATERIALIZED VIEW view_name
AS
SELECT column1, column2, ...
FROM table_name
WHERE conditions
[WITH [NO] DATA];
```

With data: It is optional tells PostgreSQL to **populate(Add data in MV or Fill the Data) the materialized view immediately** with the results of the query.

With No Data: Creates the MV structure but doesn't fill it yet; you would need to run REFRESH MATERIALIZED VIEW sales_summary

Syntax: REFRESH MATERIALIZED VIEW sales_summary;

Question:

1. Create the necessary database schema

Create three tables:

- client_master
- service_catalog
- invoice_details

Insert sample data for clients, services, and invoices.

2. Create a View

Create a view vw_invoice_summary that shows:

- **invoice_id**
- **invoice_date**
- **client_name**
- **service_name**
- **final_amount = (hourly_rate × hours_worked) − discount**

3. Implement Role-Based Access

1. Create a role **ARJUN** with login capability
2. Grant **SELECT** permission only on the view
3. Verify access by switching to role ARJUN
4. Confirm that ARJUN **cannot** access base tables
5. Revoke ARJUN's access and verify that the view becomes inaccessible

Table: client_master

client_id	client_name	email
1	Rahul Sharma	rahul@gmail.com
2	Anita Verma	anita@gmail.com
3	Karan Singh	karan@gmail.com

Table: service_catalog

service_id	service_name	hourly_rate
101	Web Development	1200.00
102	Data Analysis	1500.00
103	SEO Optimization	800.00

Table: invoice_details

invoice_id	client_id	service_id	hours_worked	discount_percent	invoice_date
1001	1	101	10	10	2025-01-05
1002	2	102	8	5	2025-01-07
1003	3	103	12	0	2025-01-10

Referential Integrity

If Table A depends on Table B, then A cannot contain invalid or non-existing references.

CASCADE in SQL:

CASCADE is an action used with **FOREIGN KEY constraints** to control what happens to child-table rows when a referenced row in the parent table is **UPDATED** or **DELETED**.

Action	Meaning
ON DELETE CASCADE	Delete child rows automatically
ON UPDATE CASCADE	Update child rows automatically
ON DELETE SET NULL	Make child FK NULL
SET DEFAULT	Set child FK to default
NO ACTION / RESTRICT	Prevent delete/update if child exists

ON DELETE SET DEFAULT

If a parent row is deleted → child rows get a **default value**.

Requirement: Child column must have a **DEFAULT** value defined.

```
CREATE TABLE department (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);

CREATE TABLE employee (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    dept_id INT NOT NULL DEFAULT 1,
    FOREIGN KEY (dept_id)
        REFERENCES department(dept_id)
        ON DELETE SET DEFAULT
);
```

All employees with dept_id = 10 will automatically get dept_id = 1 (the default).

ON DELETE SET NULL

If a row in the parent table is deleted → the foreign key in the child table becomes **NULL**.

```sql
CREATE TABLE department (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);

CREATE TABLE employee (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    dept_id INT NULL,
    FOREIGN KEY (dept_id)
        REFERENCES department(dept_id)
        ON DELETE SET NULL
);
```

All employees with dept_id = 10 will have `dept_id = NULL`.

PS 02

- You are tasked with designing a database system to manage customer data and their associated purchase records. The goal is to ensure that every order in the **Orders** table is correctly linked to a valid customer in the **Customer** table through a **foreign key constraint** on the **C_id** column. The solution should:

1. Prevent insertion of orders with non-existent **C_id** values.
2. Automatically handle updates or deletions in the **Customer** table to maintain referential integrity.
3. Provide insights into the impact of implementing foreign key constraints on database operations like cascading updates and deletions.

Winter Winning Camp

DATABASE MANAGEMENT SYSTEM

Department Of Career Planning & Development

Delete Duplicate Emails

Ques 1:

You are given a table Person containing id and email.

Each email may appear multiple times, but you must delete all duplicate emails, keeping only the row with the smallest id for each email.

Write a DELETE statement that removes duplicate rows.

The final table should contain only unique emails.

Person Table

id	email
1	john@example.com
2	bob@example.com
3	john@example.com

Output:

id	email
1	john@example.com
2	bob@example.com

Average Time of Process per Machine

Ques 2: You are given an **Activity** table that records when each process on each machine starts and ends.
Each process has exactly one start and one end entry, and start always occurs before end.

- Your task is to **calculate the average processing time** for each machine.
  A process's processing time = end.timestamp – start.timestamp.
- Return one row per machine with machine_id, processing_time (rounded to **3 decimal places**)

Activity:

machine_id	process_id	activity_type	timestamp
0	0	start	0.712
0	0	end	1.520
0	1	start	3.140
0	1	end	4.120
1	0	start	0.550
1	0	end	1.550
1	1	start	0.430
1	1	end	1.420
2	0	start	4.100
2	0	end	4.512
2	1	start	2.500
2	1	end	5.000

Output:

machine_id	processing_time
0	0.894
1	0.995
2	1.456

Winter Winning Camp

DATABASE MANAGEMENT SYSTEM

Department Of Career Planning & Development

List the Products Ordered in a Period

Ques 1 You are given two tables:

- **Products**: product details
- **Orders**: units ordered on different dates

Your task:

Find **product names** whose **total ordered units in February 2020 (2020-02-01 to 2020-02-29)** are **at least 100**, and also show their **total units**.

Products Table:

product_id	product_name	product_category
1	Leetcode Solutions	Book
2	Jewels of Stringology	Book
3	HP	Laptop
4	Lenovo	Laptop
5	Leetcode Kit	T-shirt

Orders:

product_id	order_date	unit
1	2020-02-05	60
1	2020-02-10	70
2	2020-01-18	30
2	2020-02-11	80
3	2020-02-17	2
3	2020-02-24	3
4	2020-03-01	20
4	2020-03-04	30
4	2020-03-04	60
5	2020-02-25	50
5	2020-02-27	50
5	2020-03-01	50

Output:

product_name	unit
Leetcode Solutions	130
Leetcode Kit	100

Exclusive Join

An **exclusive join** returns rows from **one table that have no match in another table**.

Or

“Give me all rows in Table A that **don’t exist** in Table B.”

Ques 2:

- Write an SQL query to find **students who are not enrolled in any course**.

Student Table:

student_id	student_name
1	Alice
2	Bob
3	Charlie
4	David

Enrollments:

student_id	course
1	DBMS
3	Java
5	Python

Output:

student_id	student_name
2	Bob
4	David

Winter Winning Camp

DATABASE MANAGEMENT SYSTEM

Department Of Career Planning & Development

Students and Examinations

Ques 1: You are given:

(1) A list of **students**

(2) A list of **subjects**

An **examinations** table that records each time a student attended an exam (duplicates allowed)

Every student takes **every** subject, but the examinations table may have multiple entries per exam or none.

Your task:

Create a table showing **for every student - subject pair**, how many times they attended that subject's exam.

Sort by **student_id**, then **subject_name**.

(Note :Output can be in Random Order)

Students Table:

student_id	student_name
1	Alice
2	Bob
13	John
6	Alex

Subjects Table:

subject_name
Math
Physics
Programming

Examination Table

student_id	subject_name
1	Math
1	Physics
1	Programming
2	Programming
1	Physics
1	Math
13	Math
13	Programming
13	Physics
2	Math
1	Math

Output:

student_id	student_name	subject_name	attended_exams
1	Alice	Math	3
1	Alice	Physics	2
1	Alice	Programming	1
2	Bob	Math	1
2	Bob	Physics	0
2	Bob	Programming	1
6	Alex	Math	0
6	Alex	Physics	0
6	Alex	Programming	0
13	John	Math	1
13	John	Physics	1
13	John	Programming	1

DATE DIFFERENCE:

(i) DATE DIFFERENCE IN DAYS (INTEGER)

```
--DATE DIFFERENCE TO FIND THE THE NUMBER OF DAYS BETWEEN TWO DATE
SELECT DATE(NOW())-'20-11-2023'::DATE as days
```

(ii) Difference in Months (INTEGER)

First Understand DATE_PART() :

In PostgreSQL, **DATE_PART()** is a function used to extract a specific part of a **date**, **timestamp**, or **interval**.

Syntax:

```
DATE_PART('field', source)
```

field → The part you want (string)

Example: 'year', 'month', 'day', 'hour', 'minute', 'second'.

```
SELECT DATE_PART('month', NOW());

SELECT DATE_PART('year', NOW());

SELECT DATE_PART('day', NOW());
```

Field	Meaning
year	Year number
month	1–12
day	Day of month(Number)
hour	0–23
minute	0–59
second	0–59
dow	Day of week (0=Sunday)
doy	Day of year (1–365/366)
epoch	Seconds since 1970

Now understand Date Function

Age() Function :It give date difference with year month days

(1)

```
1
2   -- IT GIVES YOU DIFFERENCE OUTPUT-> "1 year 11 mons 30 days"
3   SELECT AGE('2025-01-01', '2023-01-02');
```

(2)

```
--OUTPUT: "3 mons 30 days"
SELECT AGE('2025-02-20','2024-10-21');
```

From	To
2024-10-21 → 2024-11-21	1 month
2024-11-21 → 2024-12-21	1 month
2024-12-21 → 2025-01-21	1 month
2025-01-21 → 2025-02-21	1 month

(b) **Now difference in Month** and below gives you output 4:

```
SELECT ((DATE_PART('year', AGE('2025-02-20','2024-10-20')))*12)+
        DATE_PART('month',AGE('2025-02-20','2024-10-20')) AS DIFF_MONTHS
```

(b) **Difference in month**

Below gives you **output: 3 (It is a month)**

```
SELECT ((DATE_PART('year', AGE('2025-02-20','2024-10-21')))*12)+
        DATE_PART('month',AGE('2025-02-20','2024-10-21')) AS DIFF_MONTHS
```

(iii) Difference in Year between two dates

(a)

```
-- (iii) Difference in year between two dates

   SELECT DATE_PART('year',AGE('2025-02-20','2024-02-21'))
```

Output : 0 because 1 year complete from ‘2024-02-21’ to ‘2025-02-21’

(b)

Output: 1

```
-- (iii) Difference in year between two dates

   SELECT DATE_PART('year',AGE('2025-02-21','2024-02-21'))
```

To_char() Function:

(i) Extract Year

```
--EXTRACT YEAR
SELECT TO_CHAR('2019-01-07'::DATE,'yyyy')
```

(ii) Extract Year With Month

```
--EXTRACT MONTH WITH YEAR
SELECT TO_CHAR('2019-01-07'::DATE,'yyyy-mm')
```

<u>In SQL SERVER AND MYSQL:</u> All are small in MYSQL (For year also)

```
-- SQL SERVER
SELECT FORMAT(GETDATE(), 'yyyy-MM');
-- In MYSQL
SELECT DATE_FORMAT(order_date, '%Y-%m')
```

Monthly Transactions

Ques 2 : You are given a table of transactions including:

(1) country

(2) state (approved/declined)

(3) amount

(4) trans_date

For each **month** (YYYY-MM) and **country**, you must compute:

=> trans_count → total number of transactions

=>**approved_count** → number of approved transactions

=> **trans_total_amount** → sum of all transaction amounts

=> approved_total_amount → sum of approved transaction amounts

Return the results in any order.

Transactions Table

id	country	state	amount	trans_date
121	US	approved	1000	2018-12-18
122	US	declined	2000	2018-12-19
123	US	approved	2000	2019-01-01
124	DE	approved	2000	2019-01-07

Output:

month	country	trans_count	approved_count	trans_total_amount	approved_total_amount
2018-12	US	2	1	3000	1000
2019-01	US	1	1	2000	2000
2019-01	DE	1	1	2000	2000

Winter Winning Camp

DATABASE MANAGEMENT SYSTEM

Department Of Career Planning & Development

Ques 1:

We have a Table Employees , Department Write a query to show for each department:

(i) Department Name

(ii) Number of employees in that department

(iii) Average Salary

(iv) Salary Category (based on average salary)

- >= 60000 → "High Avg"
- 50000–59999 → "Medium Avg"
- < 50000 → "Low Avg“
- If there is no Employee in a Department-> "No Employees"

Departments Table:

DeptID	DeptName	Location
10	HR	Delhi
20	IT	Mumbai
30	Finance	Pune
50	Marketing	Chennai

Employees Table:

EmpID	EmpName	DeptID	Salary
1	Akash	10	50000
2	Nisha	20	60000
4	Simran	30	70000
5	Meena	30	55000

Output:

DeptName	EmployeeCount	AvgSalary	SalaryCategory
Finance	2	62500	High Avg
HR	1	50000	Medium Avg
IT	1	60000	High Avg
Marketing	0	NULL	No Employees

Ques 2

We have an Triangle table (x, y, z) is the primary key column for this table and Each row of this table contains the lengths of three line segments.

Write a query to check whether three Segments formed a triangle or not.

Input:

x	y	z
13	15	30
10	20	15

Output:

x	y	z	Triangle_formed
13	15	30	No
10	20	15	Yes

Winter Winning Camp

DATABASE MANAGEMENT SYSTEM

Department Of Career Planning & Development

Subqueries:

A **subquery** is a query inside another SQL query.
It runs first and gives output to the outer query.

Types of Subquery:

(i) **Inline subquery** : An **inline subquery** is a **SELECT query written inside another SELECT query** (usually inside the FROM clause).

- It works like a **temporary table inside the main query**.

Syntax:

```
SELECT columns
FROM (
        SELECT ...
     ) AS temp_table
WHERE ...
```

(ii) **Nested Subquery**:

Subquery inside Subquery is a **nested Subquery**

(iii) **Multi-Row Subquery :**

- A *multi-row* subquery returns **multiple rows**, and you must use operators like:
- IN
- ANY
- ALL
- EXISTS

(iv) **Correlated subquery** : A **correlated subquery** is a subquery that **depends on a value from the outer query**.

- The subquery runs **again and again** for each row of the outer query.
- It uses a column from the outer query inside the inner query.
- It is slower than non-correlated subqueries

(v) ***Scaler Subquery:*** A **scalar subquery** is a subquery that returns **exactly one value** (one row & one column).

(vi) SELF-CONTAINED SUBQUERY (NON-CORRELATED SUBQUERY) :

- A **self-contained subquery** is a subquery that **does NOT depend on the outer query**.

⇒ It can run **independently**
=>It returns a value used by the outer query
=>Outer query does NOT supply data to the inner query
=> Also called **non-correlated subquery**

Example: With in ,all ,any, find second highest salary

Table of Self-Contained Subquery

Type	Example	Dependent on Outer Query?	Self-Contained?
Subquery in WHERE	salary > (SELECT AVG)	No	Yes
Subquery in SELECT	Select name,(Select COUNT(*) from Employees)	No	Yes
Subquery in FROM	Temp table	No	Yes
IN / ANY / ALL	dept_id IN (SELECT)	No	Yes
EXISTS (if no outer reference)	EXISTS(SELECT 1)	No	Yes

Problem Statement 1:

You are given two tables:

Employees:

emp_id	name	dept_id	salary
1	Asha	10	50000
2	Rohan	10	70000
3	Meera	20	40000
4	Kabir	20	55000
5	Isha	30	90000

Departments Table

dept_id	dept_name
10	HR
20	Finance
30	IT

For each department, find the employee whose **salary is closest to the department's average salary.**

(If two employees are equally close, return the one with the HIGHER salary)

Output:

dept_name	emp_name	salary	dept_avg_salary
HR	Rohan	70000	60000
Finance	Kabir	55000	47500
IT	Isha	90000	90000

Ques 2:

You are given a table named UserProfile containing the following columns: user_ID, Name, Age, and Email.

Some users in the table may share the same email address. Your task is to identify all duplicate email addresses and update the record that has the highest ID for each duplicate email. Specifically, update the Email field of that record to the string 'duplicate email'.

ID	Name	Age	Email
1	Rajesh	28	rajesh@gmail.com
2	Priya	25	priya@gmail.com
3	Suresh	30	rajesh@gmail.com
4	Anjali	24	priya@gmail.com

Output:

ID	Name	Age	Email
1	Rajesh	28	rajesh@gmail.com
2	Priya	25	priya@gmail.com
3	Suresh	30	duplicate email
4	Anjali	24	duplicate email

Session 7

Data Normalization

1. Consider a relation R having attributes as R(ABCD), functional dependencies are given below:

AB->C, C->D, D->A

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

2. Relation R(ABCDE) having functional dependencies as :

A->D, B->A, BC->D, AC->BE

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

Data Normalization

3. Consider a relation R having attributes as R(ABCDE), functional dependencies are given below:

B->A, A->C, BC->D, AC->BE

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

4. Consider a relation R having attributes as R(ABCDEF), functional dependencies are given below:

A->BCD, BC->DE, B->D, D->A

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

Data Normalization

7. Relation R(ABCDE) having dependencies as:

CE -> D

D ->B

C -> A

Highest normal form? C.K set?

8. Relation R(ABCDEF) having dependencies as:

AB -> C

DC -> AE

E -> F

Highest normal form? C.K set?

Session 8

Data Normalization

9. Relation R(ABCDEFGHI) having dependencies as:

AB -> C

BD -> EF

AD -> GH

A -> I

Highest normal form? C.K set?

10. Relation R(ABCDE) having dependencies as:

AB -> CD

D -> A

BC -> DE

Highest normal form? C.K set?

Data Normalization

11. Consider a relation P having fields as F1, F2, F3, F4, F5 with the functional dependencies as follows:

F1 -> F3

F2 -> F4

(F1,F2) -> F5

In terms of normalization, this table is in which normal form. Describe the prime and non-prime attributes for the relation P.

Data Normalization

12. DBA of a university information management system wants to remove redundancy by considering the following dependencies and relation R(R_N, Name, Dob, Age) for the system. For better optimization of database, he needs to normalize the database by identifying the relevant CK keys and normal form.

Dob ->Age

Name ->Rno

Age -> Eligibility

Rno -> Name

C_N -> Cname

C_N -> Instructor

(R_N, C_N) -> Grade

Data Normalization

Relation:
R(A, B, C, D, E)
Functional Dependencies:
F = { A → B, B → C, AC → D, D → E }

Identify CK? And Highest Normal form?

Relation **R(A, B, C, D, E)**
FDs:
1.A → BC
2.B → D
3.CD → A
4.E → A
Identify CK? And Highest Normal form?

Relation:
R(P, Q, R, S)
Functional Dependencies:
F = { PQ → R, R → S, S → Q }

Identify CK? And Highest Normal form?

Relation **R(P, Q, R, S, T)**
FDs:
1.PQ → R
2.R → S
3.S → T
4.T → P
Identify CK? And Highest Normal form?

Session 9

Data Normalization

13. **Consider a relation R(ABCDEFGHI) having FDs as:**

AB -> C
AD -> GH
BD -> EF
A -> I
H -> J

•Identify that relation is in which normal form and the candidate key by determining the prime and non prime attributes.
•Apply the decomposition if needed to remove the redundancy.
•Identify how many new tables will be formed after the decomposition.

Data Normalization

14. **Relation R having attributes as (A, B, C, D, E) having the functional dependencies as follows:**

A -> B

B -> E

C -> D

•Identify that relation is in which normal form and the candidate key by determining the prime and non prime attributes.

•Apply the decomposition if needed to remove the redundancy.

•Identify how many new tables will be formed after the decomposition.

Data Normalization

15. Raj faces a problem in resolving the decomposition issue with the relation R having attributes as (A,B,C,D,E,F,G,H,I,J) having functional dependencies set as:

AB -> C

B -> F

F -> GH

D -> IJ

A -> DE

The task here is to determine the highest normal form of relation R by determining the prime and non-prime fields.

Check if relation is in 3NF or not. If not then, remove the redundancy and decompose the relation. Also, identify the possible number of relations after the decomposition and check if the decomposition is lossy or loss-less.

Session 10 and 11

Problem 01

Reverse Product name Category Wise and assign Id

product_id	product	category
1	LAPTOP	ELECTRONICS
2	SMARTPHONE	ELECTRONICS
3	TABLETS	ELECTRONICS
9	PRINTER	ELECTRONICS
4	HEADPHONES	ACCESSORIES
5	SMARTWATCH	ACCESSORIES
6	KEYBOARD	ACCESSORIES
7	MOUSE	ACCESSORIES
8	MONITOR	ACCESSORIES

Output:

product_id	product	category
1	PRINTER	ELECTRONICS
2	TABLETS	ELECTRONICS
3	SMARTPHONE	ELECTRONICS
4	LAPTOP	ELECTRONICS
1	MONITOR	ACCESSORIES
2	MOUSE	ACCESSORIES
3	KEYBOARD	ACCESSORIES
4	SMARTWATCH	ACCESSORIES
5	HEADPHONES	ACCESSORIES

Problem 02:

Customers

Id	Email
1	john@example.com
2	bob@example.com
3	john@example.com

OUTPUT:

Id	Email
1	john@example.com
2	bob@example.com

Write a SQL query (using CTE & Window functions only) to delete all duplicate email entries in a table named Person, keeping only unique emails based on its smallest Id.

Problem 03

EmpID	Ename	Salary
1	AA	1000
2	BB	300

EmpID	Ename	Salary
2	BB	400
3	CC	100

OUTPUT:

EmpID	Ename	Salary
1	AA	1000
2	BB	300
3	CC	100

In the given two tables, i.e., A and B, having attributes as Empid, Ename, and Salary.
Retrieve the EmpID existing more than one time in both the tables with the lowest salary by only using **ROW_NUMBER().**

Session 15 and 16

Problem 01

A flight booking system maintains a record of all seat bookings. Each seat has a unique seat_id, and its availability is recorded in the Free column.

- Free = 1 means the seat is **available**.
- Free = 0 means the seat is **occupied**.

Passengers often request **consecutive available seats**, especially when traveling in groups. To improve the seat allocation system, you must identify **continuous sequences of available seats** to facilitate group bookings.

You are given a **Flight_Seats** table with seat IDs and their availability status (Free). Your task is to **identify and print only those seat IDs that are available (Free = 1) and appear consecutively in the seat sequence**.

This will help the airline **quickly locate** consecutive free seats for passengers traveling in groups.

Problem 01

SEAT_ID	FREE_SLOT
1	1
2	0
3	1
4	0
5	1
6	1
7	1
8	0
9	1
10	1

OUTPUT:

SEAT_ID
5
6
7
9
10

Problem 02

Session_table

Session_id	Duration
1	30
2	199
3	299
4	580
5	1000

session_id is the primary key for this table.

duration is the time in seconds that a user has visited the application.

You want to know how long a user visits your application. You decided to create bins of "[0-5>", "[5-10>", "[10-15>" and "15 minutes or more" and count the number of sessions on it.

Write an SQL query to report the (bin, total) in any order.