22AIE303 – PROJECT REPORT

GROUP 5

Name	Roll number
Aparna Padma B	AM.EN.U4AIE22005
Anuvind M P	AM.EN.U4AIE22010
R S Harish Kumar	AM.EN.U4AIE22042
Siddharth Menon	AM.EN.U4AIE22048

I. Abstract

ICTS department has the provision of five printers to print five different files at a time. The staff member can request for printing the required file. The files which have been printed, the details of the person who requested the print, date and time of print is to be maintained. Each print job is assigned to a clerical staff to verify if the printout request is valid, the staff who requested it is eligible, whether the request is official or personal, etc. based on which he/she takes the printout. One delivery person is responsible to deliver the printouts to respective staff member after checking the personal details such as name, room no of the staff member, phone no. The print details such as type of paper used (A3, A4, etc.), single page print or double side print, color print or not, etc. is also to be maintained. At a time, a file can be printed only on one printer. The cost of each print is to be maintained, to know the printing expense at the end of each month.

II. Project Overview

The project aims to build a database management system for a print job processing system in the ICTS department. The system involves multiple entities, including staff, clerical staff, delivery personnel, printers, and print requests. The primary goal is to maintain records of print requests, verify the legitimacy of requests, and ensure proper delivery of printouts to the requesting staff members.

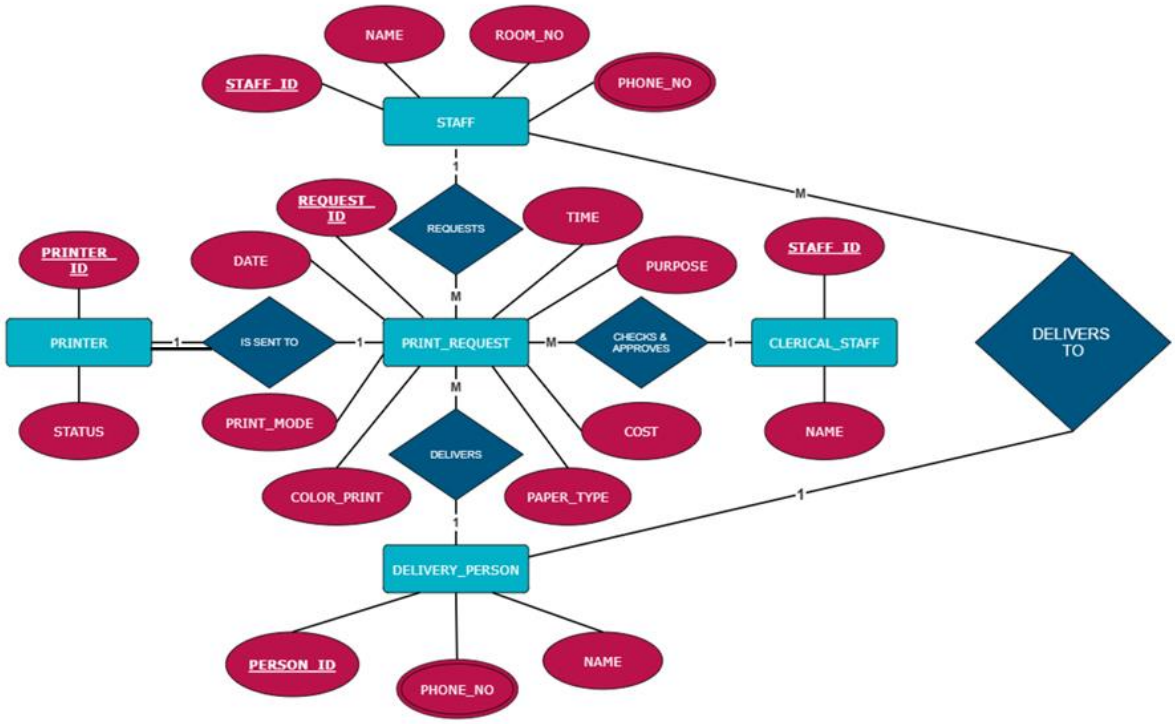
III. Phase 1: ER Diagram

The system maintains records for:

- **Printers**: A printer can handle multiple print requests at a time but can print only one file at a time.
- **Staff**: Staff members can request print jobs, which are tracked in the system.
- **Print Requests**: Each print job has specific details, such as date, time, purpose, print mode, color, paper type, and cost.
- **Clerical Staff:** Responsible for verifying the requests before printing.
- **Delivery Personnel**: They ensure that printouts are delivered to the appropriate staff members after verifying their details (e.g., name, room number, phone number).

The ER diagram for this phase includes relationships such as:

- **Staff to Print Request**: One staff member can make multiple print requests.
- **Print Request to Printer**: Each print request is assigned to a specific printer.
- **Print Request to Delivery Person:** A delivery person is responsible for delivering the printed material.
- **Clerical Staff to Print Request:** Clerical staff checks and approves print requests.
- **Delivery Person to Staff :** A delivery person is responsible for delivering the printed material to the corresponding staff.



IV. Phase 2: Relational Mapping

This phase focuses on creating relational tables based on the entities defined in Phase 1. Here is the relational mapping for the database:

1. ***STAFF*** (STAFF_ID , NAME, ROOM_NO, PERSON_ID(FK))

2. ***PRINT_REQUEST***(REQUEST_ID , DATE, TIME, PURPOSE, PRINT_MODE, COLOR_PRINT, PAPER_TYPE, COST, STAFF_ID(FK), PERSON_ID(FK), C.STAFF_ID(FK))

3. ***PRINTER*** (PRINTER_ID, STATUS, REQUEST_ID(FK))

4. ***DELIVERY_PERSON*** (PERSON_ID , NAME,)

5. ***CLERICAL_STAFF*** (STAFF_ID , NAME)

6. ***PHONE_NO_DELIVERY_PERSON*** (PHONE_NO, PERSON_ID)

7. ***PHONE_NO_STAFF*** (PHONE_NO, STAFF_ID)

Additional Phone Number Tables for both staff and delivery personnel are created to handle the multivalued attributes of phone numbers:

- Phone_No_Staff: Stores phone numbers associated with staff.

- Phone_No_Delivery_Person: Stores phone numbers associated with delivery personnel.

A. RELATIONSHIPS AND CARDINALITIES :

1. **Relationships:**
    - **Requests**: Links Staff with Print_Request, indicating which staff member made a request.
    - **Is_Sent_To:** Links Print_Request to Printer, indicating which printer is assigned to process the request.
    - **Delivers:** Links Print_Request to Delivery_Person, indicating the delivery assignment for each request.
    - **Checks & Approves:** Links Clerical_Staff to Print_Request, indicating the approval process for requests.
    - **Delivers_To:** Links Delivery_Person to Staff to show delivery completion.
2. **Cardinalities:**
    - **One** Staff can make **many** Print_Requests.
    - **One** Print_Request is sent to a **single** Printer.
    - **One** Delivery_Person can deliver **many** print jobs, and **one** print request is delivered by a **single** person.
    - **One** Clerical_Staff can approve **many** requests**.**

B. RELATIONAL SCHEMA DESCRIPTION :

1. **Staff Table**:
    - **Schema**: STAFF(STAFF_ID(PK), NAME, ROOM_NO)
    - Description: Contains details of employees requesting print jobs. The STAFF_ID is the primary key and links to Print_Request and Clerical_Staff.
2. **Print_Request Table**:
    - **Schema**: PRINT_REQUEST(REQUEST_ID(PK), DATE, TIME, PURPOSE, PRINT_MODE, COLOR_PRINT, PAPER_TYPE, COST, STAFF_ID(FK), PRINTER_ID(FK), PERSON_ID(FK))

- Description: Stores information about print requests. It has foreign keys linking to the Staff, Printer, and Delivery_Person tables.

3. **Printer Table**:
   - **Schema**: PRINTER(PRINTER_ID(PK), STATUS)
   - Description: Contains details about printers. It has a foreign key REQUEST_ID linking to the Print_Request table.
4. **Delivery_Person Table**:
   - **Schema**: DELIVERY_PERSON(PERSON_ID(PK), NAME)
   - Description: Represents delivery personnel assigned to print jobs.
5. **Clerical_Staff Table**:
   - **Schema**: CLERICAL_STAFF(STAFF_ID(PK), NAME)
   - Description: Stores information about staff members responsible for approving print requests. It shares STAFF_ID as a primary key.
6. **Phone_No_Delivery_Person Table**:
   - **Schema**: PHONE_NO_DELIVERY_PERSON(PHONE_NO, PERSON_ID(FK))
   - Description: Manages contact numbers for delivery personnel.
7. **Phone_No_Staff Table**:
   - **Schema**: PHONE_NO_STAFF(PHONE_NO, STAFF_ID(FK))
   - Description: Stores contact numbers for staff members.

V. Phase 3: Normalization

All attributes from the ER diagram is put in a one big relation to be normalized.

A. Anomalies and the Need for Normalization

1. **Insertion Anomaly:**

- Adding a new printer requires entering null values for unrelated attributes (e.g., delivery details) because all data is stored in one large table.

2. **Update Anomaly:**
   - If the room number of a staff member changes, it must be updated in multiple rows. Any oversight leads to data inconsistency.
3. **Deletion Anomaly:**
   - Deleting information about a print request might inadvertently remove critical details about a staff member or printer if stored in the same table.

B. Normalization is essential to:

- Eliminate redundancy.
- Ensure data integrity by organizing attributes into appropriate tables.
- Reduce the risk of anomalies by creating dependencies that adhere to database normalization principles.

Step 1: 1NF (First Normal Form)

- In 1NF, a relation must not contain multi-valued attributes. All attributes must have atomic values.
- In the original relation, staff_phone and delivery_person_phone are multi-valued attributes.

⇨ **To transform to 1NF:**

- We need to make staff_phone and delivery_person_phone atomic (i.e., no multiple values for a single attribute in a tuple).

⇨ **Transformation:**

- **staff_phone**: Each staff member can have multiple phones, so create a new relation for staff_phone.
- **delivery_person_phone**: Similarly, create a new relation for delivery_person_phone.

⇨ **The relations in 1NF are:**

1. **Staff**(staff_id, staff_name, staff_room_no)
   - This stores information about staff members.

2. **Staff_Phone**(staff_id, staff_phone)
    - Since a staff member can have multiple phone numbers, we separate this into a new relation.
3. **Printer**(printer_id, printer_status)
    - This stores information about printers.
4. **Clerical_Staff**(clerical_staff_id, clerk_name)
    - Information about clerical staff.
5. **Delivery_Person**(delivery_person_id, delivery_person_name)
    - Information about delivery persons.
6. **Delivery_Phone**(delivery_person_id, delivery_person_phone)
    - Since a delivery person can have multiple phone numbers, we create a separate relation for this.
7. **Print_Request**(print_request_id, staff_id, printer_id, purpose, time, date, print_mode, color_print, paper_type, cost)
    - Stores information about print requests made by staff, using a printer, with various print settings and associated costs.

Step 2: 2NF (Second Normal Form)

2NF Requirements:

- The relation must be in 1NF.
- All non-prime attributes must be fully functionally dependent on the whole primary key. In other words, we need to eliminate partial dependencies, where a non-prime attribute depends only on part of a composite primary key.

Analysis of Partial Dependencies:

- **Print_Request** has a **composite key** (print_request_id, staff_id, printer_id).
- The **partial dependency** occurs because staff_name and staff_room_no depend only on staff_id and not on the whole composite key of Print_Request. Similarly, printer_status depends only on printer_id.

Decomposition for 2NF:

- Create a new relation for staff attributes (staff_name, staff_room_no) that depend only on staff_id.
- Create a new relation for printer attributes (printer_status) that depend only on printer_id

New Relations in 2NF:

1. **Staff**(staff_id, staff_name, staff_room_no) — *No partial dependency, staff_name and staff_room_no depend fully on staff_id.*
2. **Staff_Phone**(staff_id, staff_phone) — *No partial dependency, each staff_id can have multiple staff_phone numbers.*
3. **Printer**(printer_id, printer_status) — *No partial dependency, printer_status depends fully on printer_id.*
4. **Clerical_Staff**(clerical_staff_id, clerk_name) — *This relation remains unchanged as clerk_name depends fully on clerical_staff_id.*
5. **Delivery_Person**(delivery_person_id, delivery_person_name) — *No partial dependency as delivery_person_name depends fully on delivery_person_id.*
6. **Delivery_Phone**(delivery_person_id, delivery_person_phone)
7. **Print_Request**(print_request_id, staff_id, printer_id, purpose, time, date, print_mode, color_print, paper_type, cost) — *Now this only contains attributes that depend on the whole composite key (print_request_id, staff_id, printer_id).*

Step 3: Convert to 3NF (Third Normal Form)

3NF Requirements:

- The relation must be in **2NF**.
- There should be **no transitive dependencies**, i.e., non-prime attributes must not depend on other non-prime attributes.

Analysis of Transitive Dependencies:

- In **Print_Request**, the **attribute cost** might depend on printer_id (since the cost can vary based on the printer used), and printer_id is part of the composite

key. This creates a **transitive dependency**: cost depends on printer_id, which is part of the primary key, but cost indirectly depends on the print_request_id through printer_id.

We need to remove this transitive dependency by creating a new relation for printer_id and cost.

Decomposition for 3NF:

- Create a new relation Printer_Cost that contains printer_id and cost.

New Relations in 3NF:

1. **Staff**(staff_id, staff_name, staff_room_no) — No transitive dependency.
2. **Staff_Phone**(staff_id, staff_phone) — No transitive dependency.
3. **Printer**(printer_id, printer_status) — No transitive dependency.
4. **Clerical_Staff**(clerical_staff_id, clerk_name) — No transitive dependency.
5. **Delivery_Person**(delivery_person_id, delivery_person_name) — No transitive dependency.
6. **Delivery_Phone**(delivery_person_id, delivery_person_phone) — No transitive dependency.
7. **Print_Request**(print_request_id, staff_id, printer_id, purpose, time, date, print_mode, color_print, paper_type) — No transitive dependency for cost.

8. **Printer_Cost**(printer_id, cost) — The cost attribute is now fully dependent on printer_id

Final Relations in 3NF:

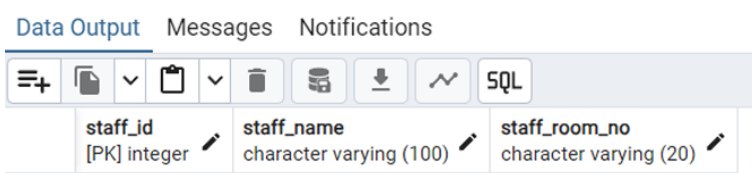
1. **Staff**(staff_id, staff_name, staff_room_no)
2. **Staff_Phone**(staff_id, staff_phone)
3. **Printer**(printer_id, printer_status)
4. **Clerical_Staff**(clerical_staff_id, clerk_name)
5. **Delivery_Person**(delivery_person_id, delivery_person_name)
6. **Delivery_Phone**(delivery_person_id, delivery_person_phone)
7. **Print_Request**(print_request_id, staff_id, printer_id, purpose, time, date, print_mode, color_print, paper_type)
8. **Printer_Cost**(printer_id, cost)

VI. Phase 4: Table Creation, Dummy Data Insertion, and Query Execution

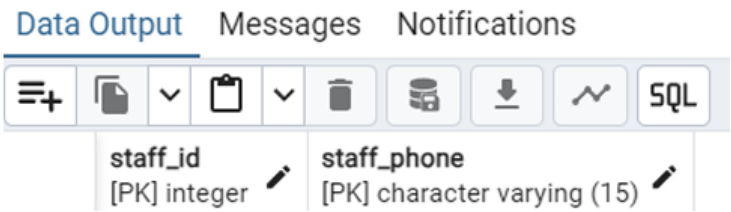
In this phase, the tables designed in Phase 3 were created using SQL. Sample data was added to these tables to test how the database works. Different queries were run to check and manage the data, showing how the system handles print job requests and related tasks.

A. Table Creation

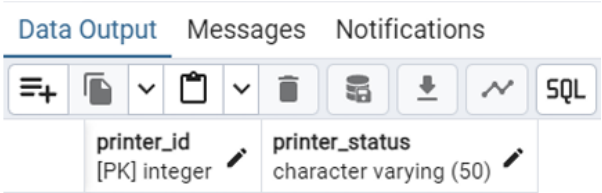
1. CREATE TABLE Staff (staff_id INT PRIMARY KEY, staff_name VARCHAR(100) NOT NULL, staff_room_no VARCHAR(20) NOT NULL);



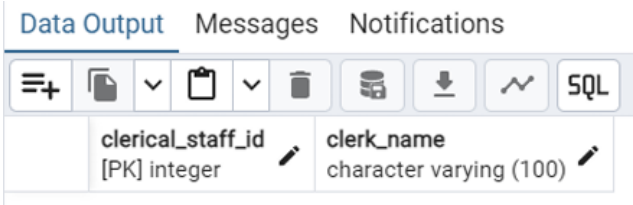
2. CREATE TABLE Staff_Phone (staff_id INT, staff_phone VARCHAR(15) NOT NULL, PRIMARY KEY (staff_id, staff_phone), FOREIGN KEY (staff_id) REFERENCES Staff(staff_id) ON DELETE CASCADE);



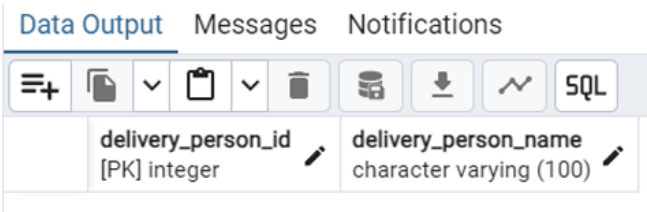
3. CREATE TABLE Printer (printer_id INT PRIMARY KEY, printer_status VARCHAR(50) NOT NULL);



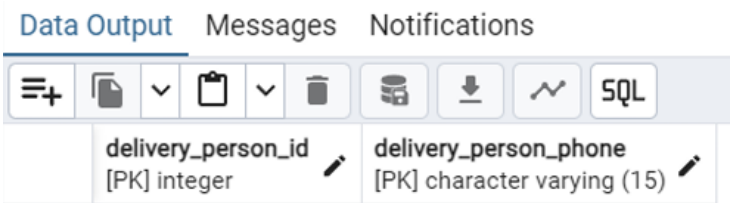
4. CREATE TABLE Clerical_Staff (clerical_staff_id INT PRIMARY KEY, clerk_name VARCHAR(100) NOT NULL);



5. CREATE TABLE Delivery_Person (delivery_person_id INT PRIMARY KEY, delivery_person_name VARCHAR(100) NOT NULL);



6. CREATE TABLE Delivery_Phone (delivery_person_id INT, delivery_person_phone VARCHAR(15) NOT NULL, PRIMARY KEY (delivery_person_id, delivery_person_phone), FOREIGN KEY (delivery_person_id) REFERENCES Delivery_Person(delivery_person_id) ON DELETE CASCADE);



7. CREATE TABLE Print_Request (print_request_id INT PRIMARY KEY, staff_id INT, printer_id INT, purpose VARCHAR(255), time TIME NOT NULL, date DATE NOT NULL, print_mode VARCHAR(50), color_print BOOLEAN, paper_type VARCHAR(50), FOREIGN KEY (staff_id) REFERENCES Staff(staff_id) ON DELETE SET NULL, FOREIGN KEY (printer_id) REFERENCES Printer(printer_id) ON DELETE SET NULL);

Data Output Messages Notifications

	print_request_id [PK] integer	staff_id integer	printer_id integer	purpose character varying (255)	time time without time zone	date date	print_mode character varying (50)	color_print boolean	paper_type character varying (50)

8. CREATE TABLE Printer_Cost (printer_id INT PRIMARY KEY, cost DECIMAL(10, 2) NOT NULL, FOREIGN KEY (printer_id) REFERENCES Printer(printer_id) ON DELETE CASCADE);

Data Output Messages Notifications

	printer_id [PK] integer	cost numeric (10,2)

B. Populating the tables

1. INSERT INTO Staff VALUES (1, 'Alice', '101'), (2, 'Bob', '102'), (3, 'Charlie', '103'), (4, 'David', '104'), (5, 'Eve', '105');

Data Output Messages Notifications

	staff_id [PK] integer	staff_name character varying (100)	staff_room_no character varying (20)
1	1	Alice	101
2	2	Bob	102
3	3	Charlie	103
4	4	David	104
5	5	Eve	105

2. INSERT INTO Staff_Phone VALUES (1, '1234567890'), (1, '0987654321'), (2, '2345678901'), (3, '3456789012'), (4, '4567890123');

Data Output Messages Notifications

	staff_id [PK] integer	staff_phone [PK] character varying (15)
1	1	1234567890
2	1	0987654321
3	2	2345678901
4	3	3456789012
5	4	4567890123

3. INSERT INTO Printer VALUES (1, 'Available'), (2, 'In Use'), (3, 'Out of Paper'), (4, 'Available');

Data Output Messages Notifications

	printer_id [PK] integer	printer_status character varying (50)
1	1	Available
2	2	In Use
3	3	Out of Paper
4	4	Available

4. INSERT INTO Clerical_Staff VALUES (1, 'Fiona'), (2, 'George');

Data Output Messages Notifications

	clerical_staff_id [PK] integer	clerk_name character varying (100)
1	1	Fiona
2	2	George

5. INSERT INTO Delivery_Person VALUES (1, 'Hannah'), (2, 'Ian');

Data Output Messages Notifications

	delivery_person_id [PK] integer	delivery_person_name character varying (100)
1	1	Hannah
2	2	Ian

6. INSERT INTO Delivery_Phone VALUES (1, '5678901234'), (1, '6789012345'), (2, '7890123456');

Data Output Messages Notifications

	delivery_person_id [PK] integer	delivery_person_phone [PK] character varying (15)
1	1	5678901234
2	1	6789012345
3	2	7890123456

7. INSERT INTO Print_Request VALUES (1, 1, 1, 'Project Report', '10:30:00', '2024-12-01', 'Single-sided', TRUE, 'A4'), (2, 2, 2, 'Invoices', '11:00:00', '2024-12-02', 'Double-sided', FALSE, 'A3'), (3, 3, 3, 'Flyers', '12:00:00', '2024-12-03', 'Double-sided', TRUE, 'Letter'), (4, 4, 4, 'Manual', '13:30:00', '2024-12-04', 'Single-sided', FALSE, 'Legal');

Data Output Messages Notifications

	print_request_id [PK] integer	staff_id integer	printer_id integer	purpose character varying (255)	time time without time zone	date date	print_mode character varying (50)	color_print boolean	paper_type character varying (50)
1	1	1	1	Project Report	10:30:00	2024-12-01	Single-sided	true	A4
2	2	2	2	Invoices	11:00:00	2024-12-02	Double-sided	false	A3
3	3	3	3	Flyers	12:00:00	2024-12-03	Double-sided	true	Letter
4	4	4	4	Manual	13:30:00	2024-12-04	Single-sided	false	Legal

8. INSERT INTO Printer_Cost VALUES (1, 5.00), (2, 7.50), (3, 3.75), (4, 6.00);

Data Output Messages Notifications

	printer_id [PK] integer	cost numeric (10,2)
1	1	5.00
2	2	7.50
3	3	3.75
4	4	6.00

C. Query Execution

1. Group by... having

Use case: Calculate the total number of print requests handled by each printer, filtering out printers with less than 2 requests.

Code:

```
SELECT printer_id, COUNT(*) AS total_requests

FROM Print_Request

GROUP BY printer_id

HAVING COUNT(*) >= 1;
```

Data Output Messages Notifications

	printer_id integer	total_requests bigint
1	4	1
2	2	1
3	3	1
4	1	1

2. Order by

Use case: List all staff in alphabetical order.

Code:

```
SELECT * FROM Staff

ORDER BY staff_name;
```

Data Output Messages Notifications

	staff_id [PK] integer	staff_name character varying (100)	staff_room_no character varying (20)
1	1	Alice	101
2	2	Bob	102
3	3	Charlie	103
4	4	David	104
5	5	Eve	105

3. Join

Use case: Find the phone numbers of the staff who made print requests

Code:

```
SELECT s.staff_name, sp.staff_phone

FROM Staff s

JOIN Staff_Phone sp ON s.staff_id = sp.staff_id;
```

Data Output Messages Notifications

	staff_name character varying (100)	staff_phone character varying (15)
1	Alice	1234567890
2	Alice	0987654321
3	Bob	2345678901
4	Charlie	3456789012
5	David	4567890123

4. Aggregate functions

Use case: Find the total cost incurred for each printer.

Code:

```
SELECT p.printer_id, SUM(pc.cost) AS total_cost

FROM Printer p

JOIN Printer_Cost pc ON p.printer_id = pc.printer_id

GROUP BY p.printer_id;
```

Data Output Messages Notifications

	printer_id [PK] integer	total_cost numeric
1	4	6.00
2	2	7.50
3	3	3.75
4	1	5.00

5. Query having Boolean operators

Use case: Find all color print requests made on A4 paper.

Code:

```
SELECT *

FROM Print_Request

WHERE color_print = TRUE AND paper_type = 'A4';
```

Data Output Messages Notifications

	print_request_id [PK] integer	staff_id integer	printer_id integer	purpose character varying (255)	time time without time zone	date date	print_mode character varying (50)	color_print boolean	paper_type character varying (50)
1	1	1	1	Project Report	10:30:00	2024-12-01	Single-sided	true	A4

6. Query having arithmetic operators

Use case: Increase all printer costs by 10%.

Code:

```
SELECT printer_id, cost, cost * 1.10 AS increased_cost

FROM Printer_Cost;
```

Data Output Messages Notifications

	printer_id [PK] integer	cost numeric (10,2)	increased_cost numeric
1	1	5.00	5.5000
2	2	7.50	8.2500
3	3	3.75	4.1250
4	4	6.00	6.6000

7. Search query using string operators

Use case: Find all staff whose name starts with 'A'.

Code:

```
SELECT *

FROM Staff

WHERE staff_name LIKE 'A%';
```

Data Output Messages Notifications

	staff_id [PK] integer	staff_name character varying (100)	staff_room_no character varying (20)
1	1	Alice	101

8. Usage of to_char, extract

Use case: Extract and format the time of all print requests.

Code:

```
SELECT print_request_id, TO_CHAR(time, 'HH:MI AM') AS formatted_time, EXTRACT(HOUR
FROM time) AS hour

FROM Print_Request;
```

Data Output Messages Notifications

	print_request_id [PK] integer	formatted_time text	hour numeric
1	1	10:30 AM	10
2	2	11:00 AM	11
3	3	12:00 PM	12
4	4	01:30 PM	13

9. Between, IN, Not between, Not IN

Use case: Find all print requests made between '2024-12-02' and '2024-12-04', excluding requests made by staff 1 and 3.

Code:

```
SELECT *

FROM Print_Request

WHERE date BETWEEN '2024-12-02' AND '2024-12-04'

  AND staff_id NOT IN (1, 3);
```

Data Output Messages Notifications

	print_request_id [PK] integer	staff_id integer	printer_id integer	purpose character varying (255)	time time without time zone	date date	print_mode character varying (50)	color_print boolean	paper_type character varying (50)
1	2	2	2	Invoices	11:00:00	2024-12-02	Double-sided	false	A3
2	4	4	4	Manual	13:30:00	2024-12-04	Single-sided	false	Legal

10. Set operations

Use case: Combine lists of phone numbers for both staff and delivery persons. Code:

```
SELECT staff_phone AS
phone FROM Staff_Phone
UNION
SELECT delivery_person_phone AS
phone FROM Delivery_Phone;
```

Data Output Messages Notifications

	phone character varying (15)
1	1234567890
2	7890123456
3	4567890123
4	3456789012
5	6789012345
6	0987654321
7	2345678901
8	5678901234

VII. Conclusion

The DBMS project for the ICTS department provides a robust system for managing print requests, including the necessary steps for data normalization and relational mapping. The system ensures efficiency,

reduces redundancy, and facilitates easy retrieval and management of print job information. The tables, relationships, and normalization steps outlined above make the database design both efficient and scalable for future expansion.