**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 |

1. **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.

1. **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.

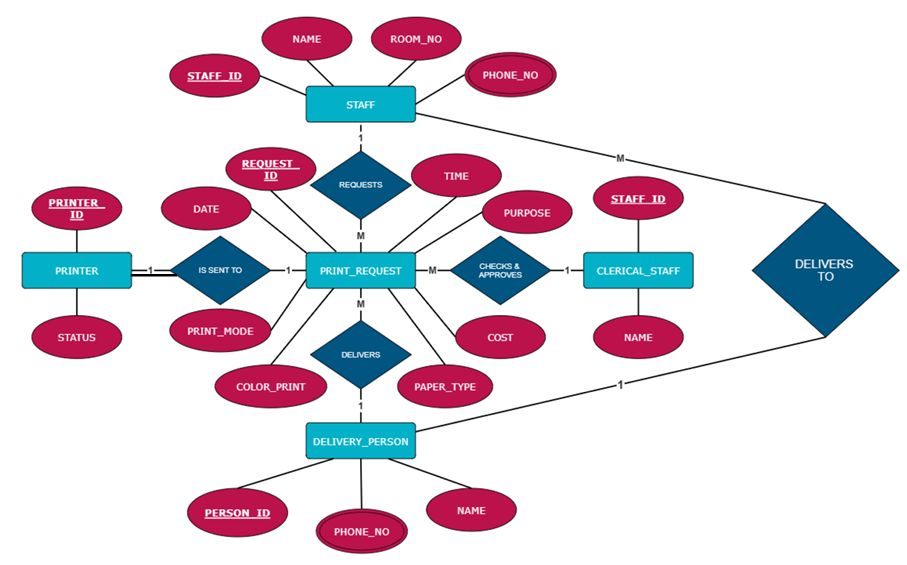
1. **Phase 1: ER Diagram**

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).

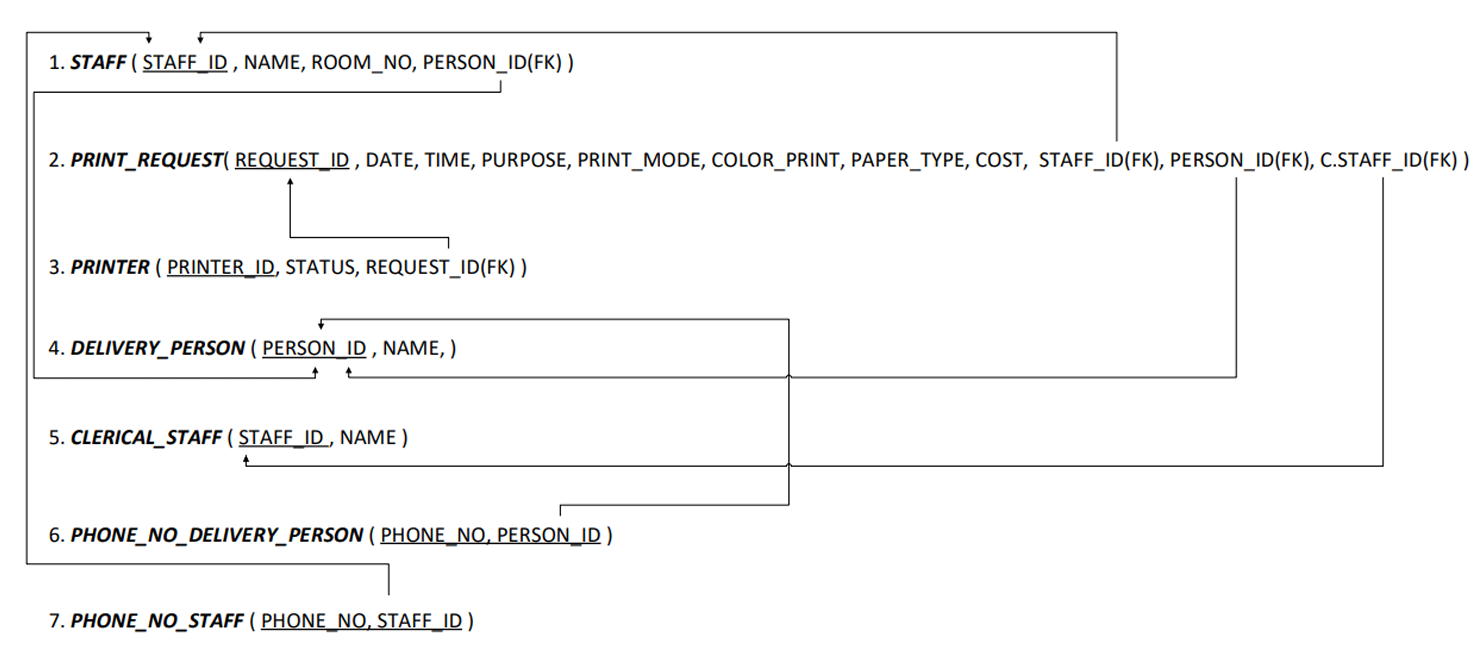
The ER diagram for this phase includes relationships such as:

* **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.



1. **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:



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.
  1. **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**.**
   1. **RELATIONAL SCHEMA DESCRIPTION :**
3. **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.
4. **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.
5. **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.
6. **Delivery\_Person Table**:
   * **Schema**: DELIVERY\_PERSON(PERSON\_ID(PK), NAME)
   * Description: Represents delivery personnel assigned to print jobs.
7. **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.
8. **Phone\_No\_Delivery\_Person Table**:
   * **Schema**: PHONE\_NO\_DELIVERY\_PERSON(PHONE\_NO, PERSON\_ID(FK))
   * Description: Manages contact numbers for delivery personnel.
9. **Phone\_No\_Staff Table**:
   * **Schema**: PHONE\_NO\_STAFF(PHONE\_NO, STAFF\_ID(FK))
   * Description: Stores contact numbers for staff members.
10. **Phase 3: Normalization**

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

* + 1. **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.
     1. **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

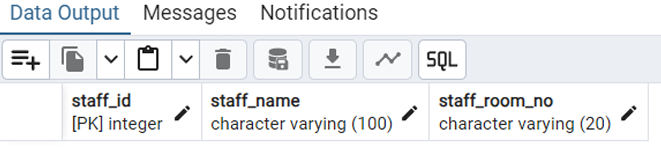
**Final Relations in 3NF:**

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)
9. **Phase 4: Table Creation, Dummy Data Insertion, and Query Execution**

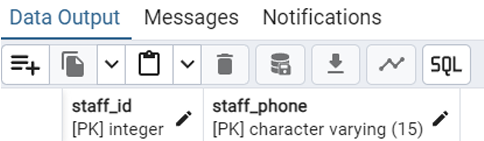
In this phase, the tables designed in Phase 3 were created using SQL. Sample data was added to these tables to test how the database works. Different queries were run to check and manage the data, showing how the system handles print job requests and related tasks.

* 1. **Table Creation**

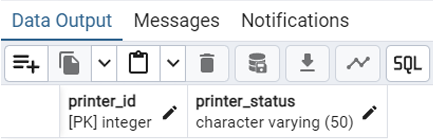
1. CREATE TABLE Staff ( staff\_id INT PRIMARY KEY, staff\_name VARCHAR(100) NOT NULL, staff\_room\_no VARCHAR(20) NOT NULL );



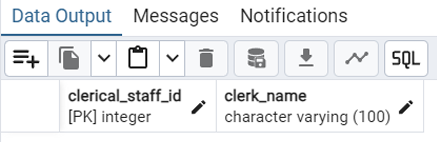
1. CREATE TABLE Staff\_Phone ( staff\_id INT, staff\_phone VARCHAR(15) NOT NULL, PRIMARY KEY (staff\_id, staff\_phone), FOREIGN KEY (staff\_id) REFERENCES Staff(staff\_id) ON DELETE CASCADE );



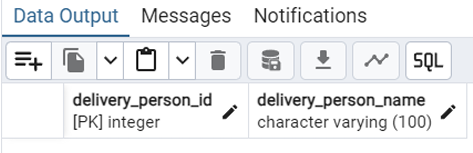
1. CREATE TABLE Printer ( printer\_id INT PRIMARY KEY, printer\_status VARCHAR(50) NOT NULL );



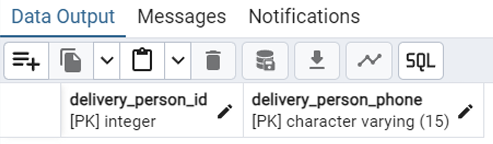
1. CREATE TABLE Clerical\_Staff ( clerical\_staff\_id INT PRIMARY KEY, clerk\_name VARCHAR(100) NOT NULL );



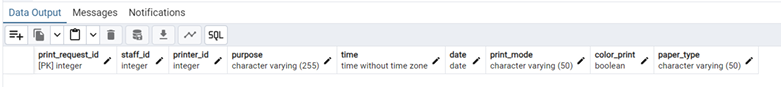
1. CREATE TABLE Delivery\_Person ( delivery\_person\_id INT PRIMARY KEY, delivery\_person\_name VARCHAR(100) NOT NULL );



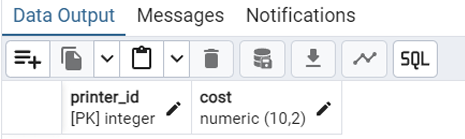
1. CREATE TABLE Delivery\_Phone ( delivery\_person\_id INT, delivery\_person\_phone VARCHAR(15) NOT NULL, PRIMARY KEY (delivery\_person\_id, delivery\_person\_phone), FOREIGN KEY (delivery\_person\_id) REFERENCES Delivery\_Person(delivery\_person\_id) ON DELETE CASCADE );



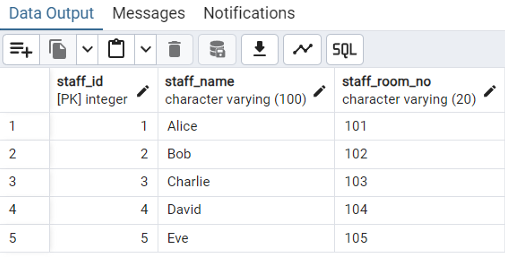
1. CREATE TABLE Print\_Request ( print\_request\_id INT PRIMARY KEY, staff\_id INT, printer\_id INT, purpose VARCHAR(255), time TIME NOT NULL, date DATE NOT NULL, print\_mode VARCHAR(50), color\_print BOOLEAN, paper\_type VARCHAR(50), FOREIGN KEY (staff\_id) REFERENCES Staff(staff\_id) ON DELETE SET NULL, FOREIGN KEY (printer\_id) REFERENCES Printer(printer\_id) ON DELETE SET NULL );



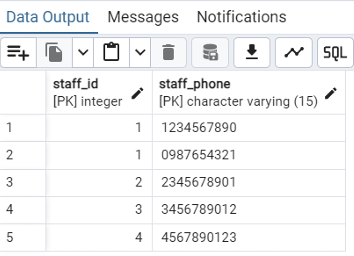
1. CREATE TABLE Printer\_Cost ( printer\_id INT PRIMARY KEY, cost DECIMAL(10, 2) NOT NULL, FOREIGN KEY (printer\_id) REFERENCES Printer(printer\_id) ON DELETE CASCADE );



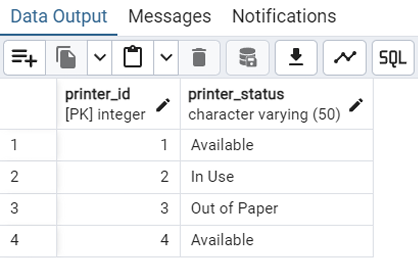
* 1. **Populating the tables**
     1. INSERT INTO Staff VALUES (1, 'Alice', '101'), (2, 'Bob', '102'), (3, 'Charlie', '103'), (4, 'David', '104'), (5, 'Eve', '105');



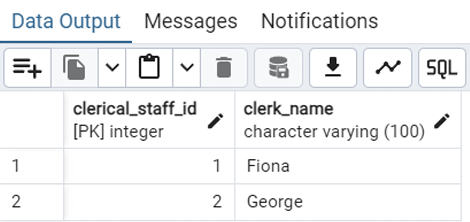
* + 1. INSERT INTO Staff\_Phone VALUES (1, '1234567890'), (1, '0987654321'), (2, '2345678901'), (3, '3456789012'), (4, '4567890123');



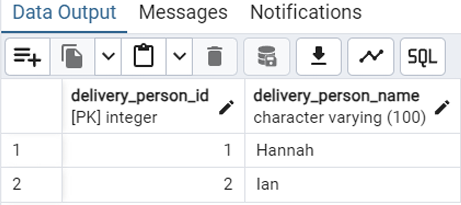
* + 1. INSERT INTO Printer VALUES (1, 'Available'), (2, 'In Use'), (3, 'Out of Paper'), (4, 'Available');



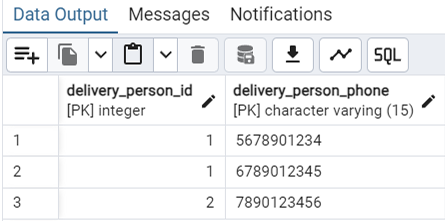
* + 1. INSERT INTO Clerical\_Staff VALUES (1, 'Fiona'), (2, 'George');



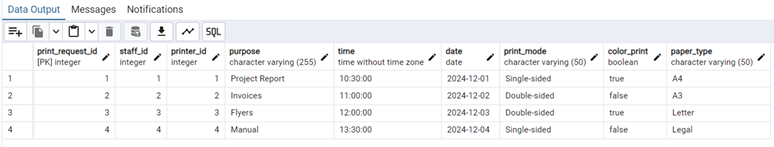
* + 1. INSERT INTO Delivery\_Person VALUES (1, 'Hannah'), (2, 'Ian');



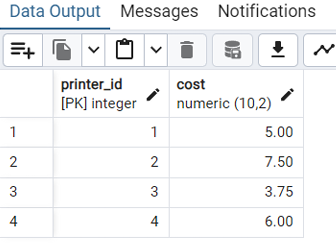
* + 1. INSERT INTO Delivery\_Phone VALUES (1, '5678901234'), (1, '6789012345'), (2, '7890123456');



* + 1. 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');



* + 1. INSERT INTO Printer\_Cost VALUES (1, 5.00), (2, 7.50), (3, 3.75), (4, 6.00);

****

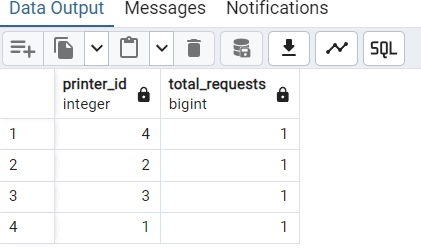
* 1. **Query Execution**

# 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

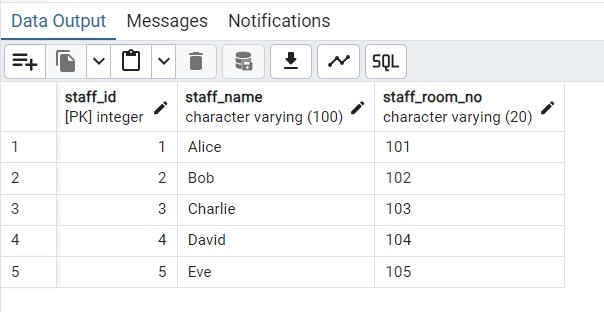
GROUP BY printer\_id HAVING COUNT(\*) >= 1;

# Order by

Use case: List all staff in alphabetical order.

Code:

SELECT \* FROM Staff ORDER BY staff\_name;

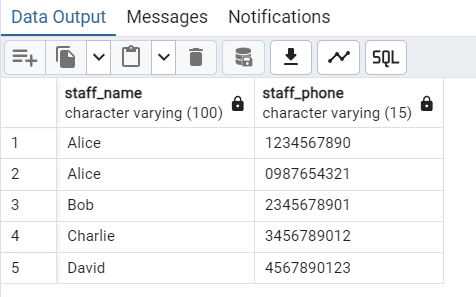


# Join

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;

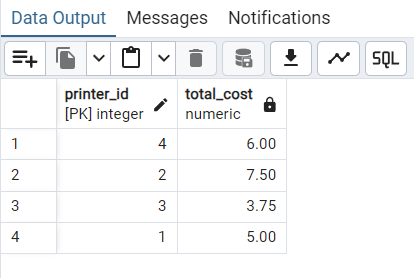


# Aggregate functions

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;

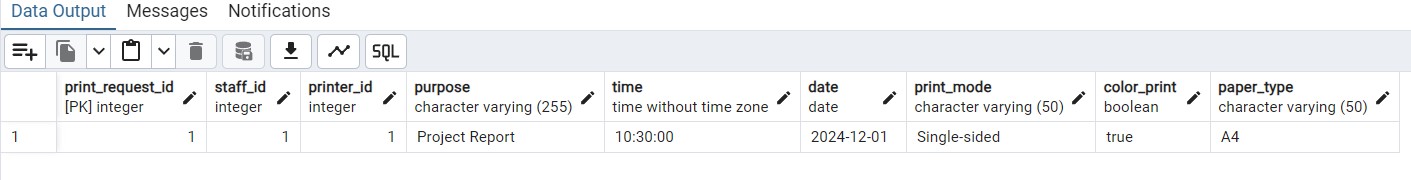


# Query having Boolean operators

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

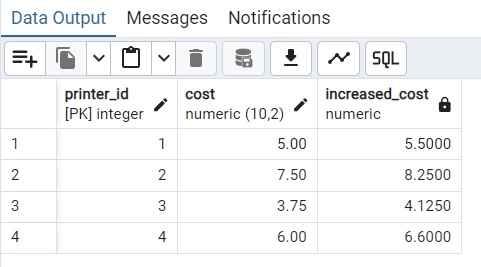
SELECT \*

FROM Print\_Request

WHERE color\_print = TRUE AND paper\_type = 'A4';

1. 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;

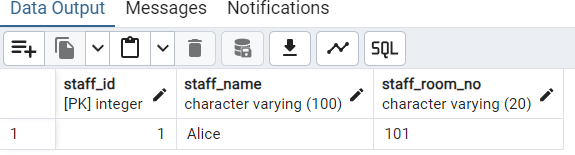


1. Search query using string operators Use case: Find all staff whose name starts with 'A'. Code:

SELECT \*

FROM Staff

WHERE staff\_name LIKE 'A%';

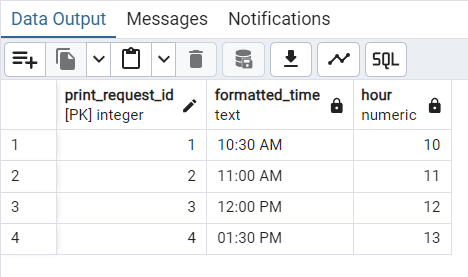


# Usage of to\_char, extract

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;



# 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.

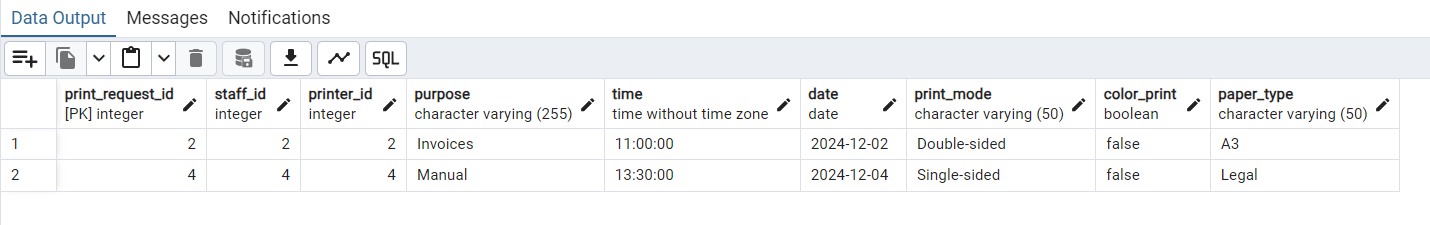
Code:

SELECT \*

FROM Print\_Request

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

AND staff\_id NOT IN (1, 3);

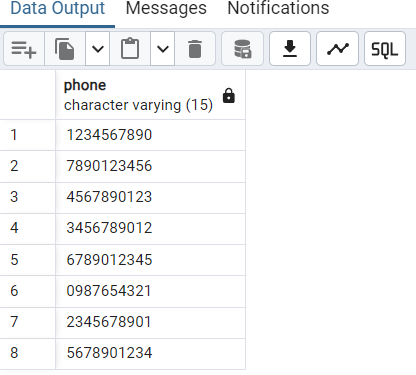


# Set operations

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;

1. **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.