PayPilot Project Report

Database Design and Implementation

**Group 1**

Aaditya Bhargav Animesh Panda Ankur Yadav Astha Singh Ayush Melty Deepank Khera Dhruv Sharma

Submitted to: Shreenath Waramballi Date: August 1, 2025

# NWG x HCLTech

1. **Problem Statement**

* Create a users table with user\_id, name, email, phone, password, pan\_details,

bank\_account\_number, ifsc\_code, and banking\_partner columns.

* Enforce constraints: email must be unique, phone must be 10 digits, and user\_id

as the primary key.

* Add address fields (city, state, pincode) with a check constraint on state to allow only Karnataka or Tamil Nadu.
* Insert five dummy users and query usernames with masked email IDs.
* Add a created\_at timestamp column with default SYSDATE.
* Query users whose names start with ’S’ and are from Bangalore.
* Design a user\_roles table to support multiple roles per user with appropriate foreign keys.

# Implementation

## Creating the Users Table

The users table is created with the specified columns and constraints to ensure data integrity.

**CREATE TABLE** users (

user\_id **VARCHAR2**(50) **PRIMARY KEY**, email **VARCHAR2**(100) **UNIQUE NOT NULL**, password **VARCHAR2**(100) **NOT NULL**,

pan\_details **VARCHAR2**(20), bank\_account\_number **VARCHAR2**(20), ifsc\_code **VARCHAR2**(11), banking\_partner **VARCHAR2**(100),

phone **VARCHAR2**(10) **CONSTRAINT** chk\_phone **CHECK**

(REGEXP\_LIKE(phone, ’^\d{10}$’)), name **VARCHAR2**(100) **NOT NULL**

);

1

2

3

4

5

6

7

8

9

10

11

Listing 1: Creating the Users Table

## Adding Address Fields

Address fields are added to the users table, with a check constraint limiting state to Kar- nataka or Tamil Nadu.

**ALTER TABLE** users **ADD** ( city **VARCHAR2**(50), state **VARCHAR2**(50),

pincode **VARCHAR2**(6),

**CONSTRAINT** chk\_state **CHECK** (state **IN** (’Karnataka’, ’Tamil Nadu’))

);

1

2

3

4

5

6

Listing 2: Adding Address Fields

## Inserting Dummy Users

Five dummy users are inserted to populate the users table.

**INSERT INTO** users (user\_id, email, password, pan\_details, bank\_account\_number, ifsc\_code, banking\_partner, phone, name, city, state, pincode)

**VALUES** (’U001’, ’[sanjay.k@example.com](mailto:sanjay.k@example.com)’, ’pass123’, ’ABCDE1234F’, ’1234567890’, ’SBIN0001234’, ’SBI’, ’9876543210’, ’Sanjay

Kumar’, ’Bangalore’, ’Karnataka’, ’560001’);

**INSERT INTO** users **VALUES** (’U002’, ’[priya.s@example.com](mailto:priya.s@example.com)’, ’pass456’, ’AIOPG0987K’, ’1111222233’, ’HDFC0005678’, ’HDFC’, ’9123456780’,

’Priya Sharma’, ’Chennai’, ’Tamil Nadu’, ’600001’);

**INSERT INTO** users **VALUES** (’U003’, ’[amit.singh@example.com](mailto:amit.singh@example.com)’, ’pass789’, ’GHYTU6789M’, ’2222333344’, ’ICIC0003456’, ’ICICI’,

’8765432109’, ’Amit Singh’, ’Mysore’, ’Karnataka’, ’570001’);

**INSERT INTO** users **VALUES** (’U004’, ’[sunita.r@example.com](mailto:sunita.r@example.com)’, ’pass321’, ’UIOPT1234N’, ’4444555566’, ’SBIN0004321’, ’SBI’,

’7654321098’, ’Sunita Rao’, ’Bangalore’, ’Karnataka’, ’560029’);

**INSERT INTO** users **VALUES** (’U005’, ’[rajesh.v@example.com](mailto:rajesh.v@example.com)’, ’pass654’, ’XYWED5678P’, ’9999888877’, ’AXIS0008765’, ’AXIS’,

’6543210987’, ’Rajesh Verma’, ’Coimbatore’, ’Tamil Nadu’, ’641001’);

1

2

3

4

5

6

7

8

9

10

Listing 3: Inserting Dummy Users

## Querying Masked Emails

A query retrieves usernames and masked email IDs, obscuring part of the email for privacy.

**SELECT** name, **SUBSTR**(email, 1, 3) || ’\*\*\*’ || **SUBSTR**(email, INSTR(email, ’@’)) **AS** masked\_email

**FROM** users;

1

2

Listing 4: Querying Masked Emails

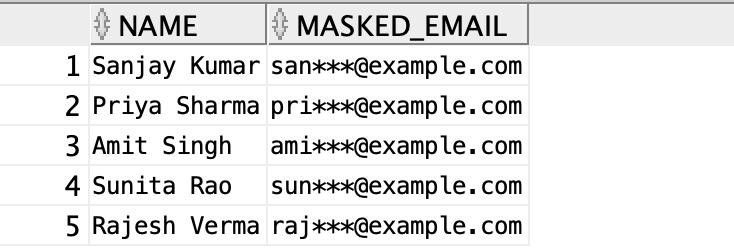


Figure 1: **Masked Email Query Output**

## Adding Created\_At Column

A created\_at column is added with a default value of SYSDATE, and existing records are updated.

**ALTER TABLE** users **ADD** created\_at **TIMESTAMP DEFAULT** SYSDATE;

**UPDATE** users **SET** created\_at = SYSDATE **WHERE** created\_at IS **NULL**;

1

2

Listing 5: Adding Created\_At Column

## Querying Users from Bangalore with Names Starting with ’S’

A query retrieves users from Bangalore whose names start with ’S’.

**SELECT** user\_id, name, city, state

**FROM** users

**WHERE** name **LIKE** ’S%’ **AND** city = ’Bangalore’;

1

2

3

Listing 6: Querying Users from Bangalore

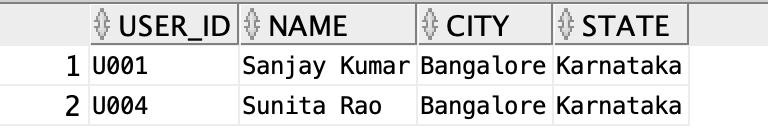


Figure 2: **Bangalore Users Query Output**

## Creating Roles and User\_Roles Tables

Tables for roles and user-role mappings are created to support multiple roles per user.

**CREATE TABLE** roles (

role\_id **NUMBER** GENERATED **BY DEFAULT AS IDENTITY PRIMARY KEY**, role\_name **VARCHAR2**(50) **UNIQUE NOT NULL**

);

**CREATE TABLE** user\_roles ( user\_id **VARCHAR2**(50), role\_id **NUMBER**,

**CONSTRAINT** pk\_user\_roles **PRIMARY KEY** (user\_id, role\_id),

**CONSTRAINT** fk\_user **FOREIGN KEY** (user\_id) REFERENCES users(user\_id) **ON DELETE CASCADE**,

**CONSTRAINT** fk\_role **FOREIGN KEY** (role\_id) REFERENCES roles(role\_id) **ON DELETE CASCADE**

);

1

2

3

4

5

6

7

8

9

10

11

12

Listing 7: Creating Roles and User\_Roles Tables

## Inserting Roles and Assigning to Users

Roles are inserted, and specific roles are assigned to users.

**INSERT INTO** roles (role\_name) **VALUES** (’Admin’); **INSERT INTO** roles (role\_name) **VALUES** (’Editor’); **INSERT INTO** roles (role\_name) **VALUES** (’Viewer’);

**INSERT INTO** user\_roles (user\_id, role\_id)

**SELECT** ’U001’, role\_id **FROM** roles **WHERE** role\_name **IN** (’Admin’, ’Editor’);

**INSERT INTO** user\_roles (user\_id, role\_id)

**SELECT** ’U002’, role\_id **FROM** roles **WHERE** role\_name = ’Viewer’;

1

2

3

4

5

6

7

8

9

Listing 8: Inserting Roles and Assignments

## Final Query: Users and Their Roles

A query joins the users, user\_roles, and roles tables to display user roles.

**SELECT** u.name, r.role\_name

**FROM** users u

**JOIN** user\_roles ur **ON** u.user\_id = ur.user\_id

**JOIN** roles r **ON** ur.role\_id = r.role\_id

**ORDER BY** u.name;

1

2

3

4

5

Listing 9: Querying Users and Roles

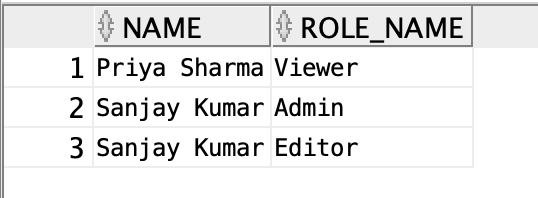


Figure 3: **User Roles Query Output**