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.

2 Implementation

2.1 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
);
```

Listing 1: Creating the Users Table

2.2 Adding Address Fields

Address fields are added to the `users` table, with a check constraint limiting `state` to Karnataka 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'))
);
```

Listing 2: Adding Address Fields

2.3 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', 'pass123', 'ABCDE1234F',
   '1234567890', 'SBIN0001234', 'SBI', '9876543210', 'Sanjay
   Kumar', 'Bangalore', 'Karnataka', '560001');

INSERT INTO users VALUES ('U002', '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',
   'pass789', 'GHYTU6789M', '2222333344', 'ICIC0003456', 'ICICI',
   '8765432109', 'Amit Singh', 'Mysore', 'Karnataka', '570001');

INSERT INTO users VALUES ('U004', 'sunita.r@example.com',
   'pass321', 'UIOPT1234N', '4444555566', 'SBIN0004321', 'SBI',
   '7654321098', 'Sunita Rao', 'Bangalore', 'Karnataka', '560029');

INSERT INTO users VALUES ('U005', 'rajesh.v@example.com',
   'pass654', 'XYWED5678P', '9999888877', 'AXIS0008765', 'AXIS',
   '6543210987', 'Rajesh Verma', 'Coimbatore', 'Tamil Nadu',
   '641001');
```

Listing 3: Inserting Dummy Users

2.4 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;
```

Listing 4: Querying Masked Emails

	NAME	MASKED_EMAIL
1	Sanjay Kumar	san***@example.com
2	Priya Sharma	pri***@example.com
3	Amit Singh	ami***@example.com
4	Sunita Rao	sun***@example.com
5	Rajesh Verma	raj***@example.com

Figure 1: **Masked Email Query Output**

2.5 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;
```

Listing 5: Adding Created_At Column

2.6 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';
```

Listing 6: Querying Users from Bangalore

	USER_ID	NAME	CITY	STATE
1	U001	Sanjay Kumar	Bangalore	Karnataka
2	U004	Sunita Rao	Bangalore	Karnataka

Figure 2: **Bangalore Users Query Output**

2.7 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
);
```

Listing 7: Creating Roles and User_Roles Tables

2.8 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';
```

Listing 8: Inserting Roles and Assignments

2.9 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;
```

Listing 9: Querying Users and Roles

	NAME	ROLE_NAME
1	Priya Sharma	Viewer
2	Sanjay Kumar	Admin
3	Sanjay Kumar	Editor

Figure 3: **User Roles Query Output**