PASSVAULT - Multi-Pass Digital Wallet

PROJECT REPORT
for
21CSC205P – DATABASE MANAGEMENT SYSTEMS

Submitted by

Saumye Singh [RA2311033010049]
Vineet Sahoo [RA2311033010053]

Under the Guidance of
Dr. S. Sadagopan
(Associate Professor, Department of Computational Intelligence)

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
WITH SPECIALIZATION IN SOFTWARE ENGINEERING

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203
FEBRUARY 2025

Project Report: PassVault - Multi-Pass Digital Wallet : (3B Component)

List of SQL Queries:

The following is a list of different types of queries applied to our database:

I) CURSOR FUNCTION

1. Cursor to Print User Emails

```
DELIMITER //
CREATE PROCEDURE Get_User_Emails()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE user_email VARCHAR(100);
   DECLARE cur CURSOR FOR SELECT Email FROM Users;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;
   read_loop: LOOP
      FETCH cur INTO user_email;
      IF done THEN
         LEAVE read_loop;
      END IF;
      SELECT user_email;
   END LOOP;
   CLOSE cur;
END;
//A
DELIMITER ;
```

2. Cursor to Update User Status Based on Transactions

```
DELIMITER //
CREATE PROCEDURE Update_User_Status()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE userId INT;
   DECLARE cur CURSOR FOR SELECT DISTINCT User_Id FROM Transactions
WHERE Transaction_Status = 'Success';
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;
   read_loop: LOOP
      FETCH cur INTO userId;
      IF done THEN
         LEAVE read_loop;
      END IF;
      UPDATE Users SET Password = 'updatedpassword' WHERE User_Id = userId;
   END LOOP;
   CLOSE cur;
END;
//
DELIMITER ;
```

3. Cursor to Count Total Transactions per User

```
DELIMITER //
CREATE PROCEDURE Count_Transactions_Per_User()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE userId INT;
   DECLARE totalTransactions INT;
   DECLARE cur CURSOR FOR SELECT User_Id, COUNT(*) FROM Transactions
GROUP BY User_Id;
```

```
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO userId, totalTransactions;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT userId, totalTransactions;
    END LOOP;
    CLOSE cur;
END;
//
DELIMITER ;
```

4. Cursor to List Users with No Transactions

```
DELIMITER //
CREATE PROCEDURE Get_Users_No_Transactions()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE user_id INT;

    DECLARE cur CURSOR FOR
        SELECT U.User_Id
        FROM Users U
        LEFT JOIN Transactions T ON U.User_Id = T.User_Id
        WHERE T.Transaction_Id IS NULL;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
```

```
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO user_id;
        IF done THEN
            LEAVE read_loop;
        END IF;
        INSERT INTO UserLogs (User_Id, Log_Message, Log_Time)
        VALUES (user_id, 'User has no transactions', NOW());
    END LOOP;

    CLOSE cur;
    SELECT ROW_COUNT() AS 'Rows Affected';
END;
//
DELIMITER ;
```

5. Cursor to Copy Old Transactions to Archive

```
DELIMITER //
CREATE PROCEDURE Archive_Old_Transactions()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE tid INT;
    DECLARE uid INT;
    DECLARE tdate DATE;
    DECLARE status VARCHAR(20);

    DECLARE cur CURSOR FOR
        SELECT Transaction_Id, User_Id, Transaction_Date, Transaction_Status
        FROM Transactions
        WHERE Transaction_Date < DATE_SUB(CURDATE(), INTERVAL 1 YEAR);
```

```
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO tid, uid, tdate, status;
        IF done THEN
            LEAVE read_loop;
        END IF;

        INSERT INTO Transactions_Archive (Transaction_Id, User_Id, Transaction_Date,
Transaction_Status)
        VALUES (tid, uid, tdate, status);

        DELETE FROM Transactions WHERE Transaction_Id = tid;
    END LOOP;

    CLOSE cur;
    SELECT ROW_COUNT() AS 'Rows Affected';
END;
//
DELIMITER ;
```

6. Cursor to Send Notifications to Users with Expired Passes

```
DELIMITER //
CREATE PROCEDURE Notify_Expired_Passes()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE user_id INT;
    DECLARE pass_name VARCHAR(255);
```

```
    DECLARE cur CURSOR FOR
        SELECT E.User_Id, P.Pass_Name
        FROM ExpirationAlerts E
        JOIN PassBankAccount P ON E.Pass_Id = P.Pass_Id
        WHERE E.Status = 'Expired';

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO user_id, pass_name;
        IF done THEN
            LEAVE read_loop;
        END IF;

        INSERT INTO Notifications (User_Id, Message, Read_Status, Timestamp)
        VALUES (user_id, CONCAT('Your pass "', pass_name, '" has expired!'), FALSE,
NOW());
    END LOOP;

    CLOSE cur;
    SELECT ROW_COUNT() AS 'Rows Affected';
END;
//
DELIMITER ;
Log
```

7. Cursor to Log Users Accessing the System

```
DELIMITER //
CREATE PROCEDURE Log_User_Access()
BEGIN
    DECLARE done INT DEFAULT FALSE;
```

```
    DECLARE user_id INT;
    DECLARE device_id VARCHAR(255);

    DECLARE cur CURSOR FOR
        SELECT User_Id, Device_Id FROM MultiDeviceAccess;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO user_id, device_id;
        IF done THEN
            LEAVE read_loop;
        END IF;

        INSERT INTO AccessLog (User_Id, Device_Id, Access_Timestamp)
        VALUES (user_id, device_id, NOW());
    END LOOP;

    CLOSE cur;
    SELECT ROW_COUNT() AS 'Rows Affected';
END;
//
DELIMITER ;
```

8. Cursor to Assign Default Preferences to New Users

```
DELIMITER //
CREATE PROCEDURE Assign_Default_Preferences()
BEGIN
    DECLARE done INT DEFAULT FALSE;
```

```
    DECLARE user_id INT;

    DECLARE cur CURSOR FOR
        SELECT User_Id FROM Users WHERE User_Id NOT IN (SELECT User_Id FROM
UserPreferences);

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO user_id;
        IF done THEN
            LEAVE read_loop;
        END IF;

        INSERT INTO UserPreferences (User_Id, Category)
        VALUES (user_id, 'General');
    END LOOP;

    CLOSE cur;
    SELECT ROW_COUNT() AS 'Rows Affected';
END;
//DELIMITER ;
```

9. Cursor to Check for Duplicate Emails

```
DELIMITER //
CREATE PROCEDURE Check_Duplicate_Emails()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE email VARCHAR(100);
    DECLARE cur CURSOR FOR
        SELECT Email FROM Users GROUP BY Email HAVING COUNT(*) > 1;
```

```
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO email;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT CONCAT('Duplicate Email Found: ', email);
    END LOOP;
    CLOSE cur;
END;
//
DELIMITER ;
```

10. Cursor to Delete Old Notifications

```
DELIMITER //
CREATE PROCEDURE Delete_Old_Notifications()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE nid INT;
    DECLARE cur CURSOR FOR
        SELECT Notification_Id FROM Notifications WHERE Timestamp <
DATE_SUB(NOW(), INTERVAL 6 MONTH);

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
```

```
        FETCH cur INTO nid;
        IF done THEN
            LEAVE read_loop;
        END IF;
        DELETE FROM Notifications WHERE Notification_Id = nid;
    END LOOP;
    CLOSE cur;
END;
//
DELIMITER ;

SHOW PROCEDURE STATUS WHERE Db = 'passvault';
```

Result:

CALL Get_User_Emails();

user_email
aarav@example.com

user_email
ananya@example.com

user_email
saumye@example.com

user_email
vineet@example.com

CALL Update_User_Status();

0 row(s) affected

CALL Count_Transactions_Per_User();

userId	totalTransactions
1	1

userId	totalTransactions
2	1

userId	totalTransactions
3	1

userId	totalTransactions
4	1

CALL Get_Users_No_Transactions();

Rows Affected
-1

CALL Archive_Old_Transactions();

Rows Affected
-1

CALL Notify_Expired_Passes();

Rows Affected
-1

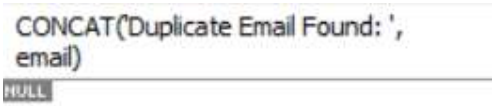
CALL Log_User_Access();

Rows Affected
-1

CALL Assign_Default_Preferences();

Rows Affected
-1

CALL Check_Duplicate_Emails();



CALL Delete_Old_Notifications();

0 row(s) affected

II) VIEW FUNCTION

1. View to Display All User Profiles

```
CREATE VIEW UserProfileView AS
SELECT u.User_Id, u.User_Name, u.Email, up.Phone_No, up.Bio, up.Profile_Picture
FROM USERS u
JOIN USERPROFILE up ON u.User_Id = up.User_Id;
```

2. View to Show Encrypted Passwords of Users

```
CREATE VIEW PasswordManagerView AS
SELECT u.User_Id, u.User_Name, pm.Encrypted_Password, pm.CreatedAt
FROM USERS u
JOIN PASSWORDMANAGER pm ON u.User_Id = pm.User_Id;
```

3. View to Display All Transactions of Users

```
CREATE VIEW TransactionsView AS
SELECT t.Transaction_Id, u.User_Name, t.Transaction_Date, t.Transaction_Status
FROM TRANSACTIONS t
JOIN USERS u ON t.User_Id = u.User_Id;
```

4. View for Active Features Available

```
CREATE VIEW ActiveFeatures AS
SELECT Feature_Id, Feature_Name
FROM FEATURES;
```

5. View for Passwords Shared Between Users

```
CREATE VIEW PasswordSharingView AS
SELECT ps.Share_Id, sender.User_Name AS Sender, receiver.User_Name AS Receiver,
ps.SharedAt, ps.Status
FROM PASSSHARING ps
JOIN USERS sender ON ps.SenderUser_Id = sender.User_Id
JOIN USERS receiver ON ps.ReceiverUser_Id = receiver.User_Id;
```

6. View for Expiring Passwords

```
CREATE VIEW ExpiringPasswords AS
SELECT ea.Alert_Id, u.User_Name, ea.Expiration_Date, ea.Status
FROM EXPIRATIONALERTS ea
JOIN USERS u ON ea.User_Id = u.User_Id
WHERE ea.Expiration_Date < DATE_ADD(CURDATE(), INTERVAL 30 DAY);
```

7. View to See All User Preferences

```
CREATE VIEW UserPreferencesView AS
SELECT u.User_Id, u.User_Name, up.Category
FROM USERPREFERENCES up
JOIN USERS u ON up.User_Id = u.User_Id;
```

8. View to Check Multi-Device Access

```
CREATE VIEW MultiDeviceAccessView AS
SELECT ma.Access_Id, ma.Device_Id, ma.Last_IP, ma.Connected_Devices
FROM MULTIDEVICEACCESS ma;
```

9. View for Backup & Recovery Status

```
CREATE VIEW BackupStatusView AS
SELECT br.Backup_Id, u.User_Name, br.Backup_Date, br.Recovery_Status,
br.Backup_Location
FROM BACKUPRECOVERY br
JOIN USERS u ON br.User_Id = u.User_Id;
```

10. View for User Notifications

```
CREATE VIEW UserNotifications AS
SELECT n.Notification_Id, u.User_Name, n.Message, n.Read_Status, n.Timestamp
FROM NOTIFICATIONS n
JOIN USERS u ON n.User_Id = u.User_Id;
```

Result:

```
SHOW FULL TABLES IN passvault WHERE Table_type = 'VIEW';
```

Tables_in_passvault	Table_type
activefeatures	VIEW
backupstatusview	VIEW
expiringpasswords	VIEW
multideviceaccessview	VIEW
passwordmanagerview	VIEW
passwordsharingview	VIEW
transactionsview	VIEW
usernotifications	VIEW
userpreferencesview	VIEW
userprofileview	VIEW

```
SELECT * FROM UserProfileView;
```

User_Id	User_Name	Email	Phone_No	Bio	Profile_Picture
1	Vineet Sahoo	vineet@example.com	9876543210	Tech enthusiast and developer	profile1.jpg
2	Saumye Singh	saumye@example.com	8765432109	Cybersecurity researcher	profile2.jpg
3	Aarav Mehta	aarav@example.com	7654321098	AI and ML enthusiast	profile3.jpg
4	Priya Sharma	priya@example.com	6543210987	Love coding and open-source	profile4.jpg
5	Rohan Verma	rohan@example.com	5432109876	Music lover and tech geek	profile5.jpg
6	Ananya Roy	ananya@example.com	4321098765	Data science enthusiast	profile6.jpg
7	Kunal Das	kunal@example.com	3210987654	Finance and investment expert	profile7.jpg
8	Shreya Iyer	shreya@example.com	9102837465	Blockchain and Web3 Developer	profile8.jpg
9	Rahul Nair	rahul@example.com	8293745610	Passionate about UI/UX design	profile9.jpg
10	Neha Kapoor	neha@example.com	7382910564	Full Stack Developer	profile10.jpg
11	Arjun Desai	arjun@example.com	6274930182	Cybersecurity and ethical hacking	profile11.jpg

SELECT * FROM PasswordManagerView;

User_Id	User_Name	Encrypted_Password	CreatedAt
1	Vineet Sahoo	encryptedpass1	2025-03-06 18:12:22
2	Saumye Singh	encryptedpass2	2025-03-06 18:12:22
3	Aarav Mehta	encryptedpass3	2025-03-06 18:12:22
4	Priya Sharma	encryptedpass4	2025-03-06 18:12:22
5	Rohan Verma	encryptedpass5	2025-03-06 18:12:22
6	Ananya Roy	encryptedpass6	2025-03-06 18:12:22
7	Kunal Das	encryptedpass7	2025-03-06 18:12:22
8	Shreya Iyer	encryptedpass8	2025-03-06 18:12:22
9	Rahul Nair	encryptedpass9	2025-03-06 18:12:22
10	Neha Kapoor	encryptedpass10	2025-03-06 18:12:22
11	Arjun Desai	encryptedpass11	2025-03-06 18:12:22

SELECT * FROM TransactionsView;

Transaction_Id	User_Name	Transaction_Date	Transaction_Status
1	Vineet Sahoo	2025-03-06 19:01:38	Success
2	Saumye Singh	2025-03-06 19:01:38	Pending
3	Aarav Mehta	2025-03-06 19:01:38	Failed
4	Priya Sharma	2025-03-06 19:01:38	Success
5	Rohan Verma	2025-03-06 19:01:38	Pending
6	Ananya Roy	2025-03-06 19:01:38	Failed
7	Kunal Das	2025-03-06 19:01:38	Success
8	Shreya Iyer	2025-03-06 19:01:38	Success
9	Rahul Nair	2025-03-06 19:01:38	Pending
10	Neha Kapoor	2025-03-06 19:01:38	Failed
11	Arjun Desai	2025-03-06 19:01:38	Success

SELECT * FROM ActiveFeatures;

Feature_Id	Feature_Name
6	AI-Based Recommendations
11	Analytics Dashboard
8	Data Backup & Recovery
5	Expiration Alerts
3	Multi-Device Acc
2	Pass Sharing
7	Pass Sync
10	Personalized Offers
1	QR Sharing
4	Transaction History
9	Two-Factor Authentication

Pass Sync

SELECT * FROM PasswordSharingView;

Share_Id	Sender	Receiver	SharedAt	Status
1	Vineet Sahoo	Saumye Singh	2025-03-06 18:14:08	Accepted
2	Saumye Singh	Aarav Mehta	2025-03-06 18:14:08	Pending
3	Aarav Mehta	Priya Sharma	2025-03-06 18:14:08	Rejected
4	Priya Sharma	Rohan Verma	2025-03-06 18:14:08	Accepted
5	Rohan Verma	Ananya Roy	2025-03-06 18:14:08	Pending
6	Ananya Roy	Kunal Das	2025-03-06 18:14:08	Rejected
7	Kunal Das	Shreya Iyer	2025-03-06 18:14:08	Accepted
8	Shreya Iyer	Rahul Nair	2025-03-06 18:14:08	Pending
9	Rahul Nair	Neha Kapoor	2025-03-06 18:14:08	Accepted
10	Neha Kapoor	Arjun Desai	2025-03-06 18:14:08	Rejected
11	Arjun Desai	Vineet Sahoo	2025-03-06 18:14:08	Accepted

SELECT * FROM ExpiringPasswords;

Alert_Id	User_Name	Expiration_Date	Status
1	Vineet Sahoo	2025-04-10	Active
2	Saumye Singh	2025-03-15	Expired

SELECT * FROM UserPreferencesView;

User_Id	User_Name	Category
1	Vineet Sahoo	Technology
2	Saumye Singh	Finance
3	Aarav Mehta	Travel
4	Priya Sharma	Health & Fitness
5	Rohan Verma	Entertainment
6	Ananya Roy	E-commerce
7	Kunal Das	Education
8	Shreya Iyer	Cybersecurity
9	Rahul Nair	Gaming
10	Neha Kapoor	Productivity
11	Arjun Desai	Investment

SELECT * FROM MultiDeviceAccessView;

Access_Id	Device_Id	Last_IP	Connected_Devices
1	Device001	192.168.1.10	Laptop, Mobile
2	Device002	192.168.1.20	Mobile
3	Device003	192.168.1.30	Tablet, Mobile
4	Device004	192.168.1.40	Laptop, Tablet
5	Device005	192.168.1.50	Smartwatch, Mobile
6	Device006	192.168.1.60	Mobile, Smart TV
7	Device007	192.168.1.70	Laptop, Mobile, Tablet
8	Device008	192.168.1.80	Workstation, Mobile
9	Device009	192.168.1.90	Laptop, Home Assistant
10	Device010	192.168.2.10	Smartphone, Wearable
11	Device011	192.168.2.20	Laptop, Smart Display

SELECT * FROM BackupStatusView;

Backup_Id	User_Name	Backup_Date	Recovery_Status	Backup_Location
1	Vineet Sahoo	2025-03-01 00:00:00	Completed	Cloud Storage
2	Saumye Singh	2025-03-02 00:00:00	Pending	Local Backup
3	Aarav Mehta	2025-03-03 00:00:00	Completed	Cloud Storage
4	Priya Sharma	2025-03-04 00:00:00	Pending	External HDD
5	Rohan Verma	2025-03-05 00:00:00	Completed	Local Backup
6	Ananya Roy	2025-03-06 00:00:00	Completed	Cloud Storage
7	Kunal Das	2025-03-07 00:00:00	Pending	USB Drive
8	Shreya Iyer	2025-03-08 00:00:00	Completed	Cloud Storage
9	Rahul Nair	2025-03-09 00:00:00	Completed	External HDD
10	Neha Kapoor	2025-03-10 00:00:00	Pending	Local Backup
11	Arjun Desai	2025-03-11 00:00:00	Completed	Cloud Storage

SELECT * FROM UserNotifications;

Notification_Id	User_Name	Message	Read_Status	Timestamp
1	Vineet Sahoo	Your pass is expiring soon!	0	2025-03-06 19:01:49
2	Saumye Singh	New security update available.	1	2025-03-06 19:01:49
12	Saumye Singh	Your pass "Concert Ticket" has expired!	0	2025-03-29 13:04:25
3	Aarav Mehta	Pass shared successfully.	0	2025-03-06 19:01:49
4	Priya Sharma	Your transaction was successful.	1	2025-03-06 19:01:49
13	Priya Sharma	Your pass "Gym Membership" has expired!	0	2025-03-29 13:04:25
5	Rohan Verma	Backup completed successfully.	0	2025-03-06 19:01:49
6	Ananya Roy	New device login detected.	1	2025-03-06 19:01:49
14	Ananya Roy	Your pass "Flight Ticket" has expired!	0	2025-03-29 13:04:25
7	Kunal Das	Pass updated successfully.	0	2025-03-06 19:01:49
8	Shreya Iyer	Upcoming event reminder!	0	2025-03-06 19:01:49
15	Shreya Iyer	Your pass "PlayStation Plus Membership" ...	0	2025-03-29 13:04:25
9	Rahul Nair	Discount coupon expires soon.	1	2025-03-06 19:01:49
10	Neha Kapoor	Password changed successfully.	0	2025-03-06 19:01:49
16	Neha Kapoor	Your pass "Tech Conference Entry" has e...	0	2025-03-29 13:04:25
11	Arjun Desai	Security alert: Unusual login detected.	1	2025-03-06 19:01:49

III) JOIN QUERIES

1. Get User Profiles with User Details (INNER JOIN)

```
SELECT U.User_Id, U.User_Name, U.Email, UP.Phone_No, UP.Bio
FROM USERS U
INNER JOIN USERPROFILE UP ON U.User_Id = UP.User_Id;
```

User_Id	User_Name	Email	Phone_No	Bio
1	Vineet Sahoo	vineet@example.com	9876543210	Tech enthusiast and developer
2	Saumye Singh	saumye@example.com	8765432109	Cybersecurity researcher
3	Aarav Mehta	aarav@example.com	7654321098	AI and ML enthusiast
4	Priya Sharma	priya@example.com	6543210987	Love coding and open-source
5	Rohan Verma	rohan@example.com	5432109876	Music lover and tech geek
6	Ananya Roy	ananya@example.com	4321098765	Data science enthusiast
7	Kunal Das	kunal@example.com	3210987654	Finance and investment expert
8	Shreya Iyer	shreya@example.com	9102837465	Blockchain and Web3 Developer
9	Rahul Nair	rahul@example.com	8293745610	Passionate about UI/UX design
10	Neha Kapoor	neha@example.com	7382910564	Full Stack Developer
11	Arjun Desai	arjun@example.com	6274930182	Cybersecurity and ethical hacking

2. List All Transactions with User Details (INNER JOIN)

```
SELECT U.User_Name, T.Transaction_Id, T.Transaction_Date, T.Transaction_Status
FROM USERS U
INNER JOIN TRANSACTIONS T ON U.User_Id = T.User_Id;
```

User_Name	Transaction_Id	Transaction_Date	Transaction_Status
Vineet Sahoo	1	2025-03-06 19:01:38	Success
Saumye Singh	2	2025-03-06 19:01:38	Pending
Aarav Mehta	3	2025-03-06 19:01:38	Failed
Priya Sharma	4	2025-03-06 19:01:38	Success
Rohan Verma	5	2025-03-06 19:01:38	Pending
Ananya Roy	6	2025-03-06 19:01:38	Failed
Kunal Das	7	2025-03-06 19:01:38	Success
Shreya Iyer	8	2025-03-06 19:01:38	Success
Rahul Nair	9	2025-03-06 19:01:38	Pending
Neha Kapoor	10	2025-03-06 19:01:38	Failed
Arjun Desai	11	2025-03-06 19:01:38	Success

3. Get Users Who Have Notifications (LEFT JOIN)

```
SELECT U.User_Name, N.Message, N.Timestamp
FROM USERS U
LEFT JOIN NOTIFICATIONS N ON U.User_Id = N.User_Id;
```

User_Name	Message	Timestamp
Vineet Sahoo	Your pass is expiring soon!	2025-03-06 19:01:49
Saumye Singh	New security update available.	2025-03-06 19:01:49
Saumye Singh	Your pass "Concert Ticket" has expired!	2025-03-29 13:04:25
Aarav Mehta	Pass shared successfully.	2025-03-06 19:01:49
Priya Sharma	Your transaction was successful.	2025-03-06 19:01:49
Priya Sharma	Your pass "Gym Membership" has expired!	2025-03-29 13:04:25
Rohan Verma	Backup completed successfully.	2025-03-06 19:01:49
Ananya Roy	New device login detected.	2025-03-06 19:01:49
Ananya Roy	Your pass "Flight Ticket" has expired!	2025-03-29 13:04:25
Kunal Das	Pass updated successfully.	2025-03-06 19:01:49
Shreya Iyer	Upcoming event reminder!	2025-03-06 19:01:49
Shreya Iyer	Your pass "PlayStation Plus Membership" ...	2025-03-29 13:04:25
Rahul Nair	Discount coupon expires soon.	2025-03-06 19:01:49
Neha Kapoor	Password changed successfully.	2025-03-06 19:01:49
Neha Kapoor	Your pass "Tech Conference Entry" has e...	2025-03-29 13:04:25
Arjun Desai	Security alert: Unusual login detected.	2025-03-06 19:01:49

4. Find Users with No Notifications (LEFT JOIN with NULL Check)

```
SELECT U.User_Name
FROM USERS U
LEFT JOIN NOTIFICATIONS N ON U.User_Id = N.User_Id
WHERE N.Notification_Id IS NOT NULL;
```

User_Name
Vineet Sahoo
Saumye Singh
Saumye Singh
Aarav Mehta
Priya Sharma
Priya Sharma
Rohan Verma
Ananya Roy
Ananya Roy
Kunal Das
Shreya Iyer
Shreya Iyer
Rahul Nair
Neha Kapoor
Neha Kapoor
Arjun Desai

5. Get Passwords Shared Between Users (INNER JOIN)

```
SELECT S.SenderUser_Id, U1.User_Name AS Sender, S.ReceiverUser_Id,
U2.User_Name AS Receiver, S.Status

FROM PASSSHARING S

INNER JOIN USERS U1 ON S.SenderUser_Id = U1.User_Id

INNER JOIN USERS U2 ON S.ReceiverUser_Id = U2.User_Id;
```

SenderUser_Id	Sender	ReceiverUser_Id	Receiver	Status
1	Vineet Sahoo	2	Saumye Singh	Accepted
2	Saumye Singh	3	Aarav Mehta	Pending
3	Aarav Mehta	4	Priya Sharma	Rejected
4	Priya Sharma	5	Rohan Verma	Accepted
5	Rohan Verma	6	Ananya Roy	Pending
6	Ananya Roy	7	Kunal Das	Rejected
7	Kunal Das	8	Shreya Iyer	Accepted
8	Shreya Iyer	9	Rahul Nair	Pending
9	Rahul Nair	10	Neha Kapoor	Accepted
10	Neha Kapoor	11	Arjun Desai	Rejected
11	Arjun Desai	1	Vineet Sahoo	Accepted

6. Get Expired Password Alerts (INNER JOIN with WHERE Condition)

```
SELECT U.User_Name, E.Expiration_Date, E.Status

FROM USERS U

INNER JOIN EXPIRATIONALERTS E ON U.User_Id = E.User_Id

WHERE E.Status = 'Expired';
```

User_Name	Expiration_Date	Status
Saumye Singh	2025-03-15	Expired
Priya Sharma	2025-07-20	Expired
Ananya Roy	2025-09-05	Expired
Shreya Iyer	2025-11-12	Expired
Neha Kapoor	2026-01-30	Expired

7. Get Users with Their Preferred Categories (LEFT JOIN)

```
SELECT U.User_Name, P.Category

FROM USERS U

LEFT JOIN USERPREFERENCES P ON U.User_Id = P.User_Id;
```

User_Name	Category
Vineet Sahoo	Technology
Saumye Singh	Finance
Aarav Mehta	Travel
Priya Sharma	Health & Fitness
Rohan Verma	Entertainment
Ananya Roy	E-commerce
Kunal Das	Education
Shreya Iyer	Cybersecurity
Rahul Nair	Gaming
Neha Kapoor	Productivity
Arjun Desai	Investment

8. List All Bank Accounts with Users (RIGHT JOIN)

```
SELECT U.User_Name, B.BankName, B.Card_No, B.Card_Type, B.Expiry_Date
FROM BANK B
RIGHT JOIN USERS U ON B.User_Id = U.User_Id;
```

User_Name	BankName	Card_No	Card_Type	Expiry_Date
Vineet Sahoo	HDFC Bank	1234567812345678	Credit	2027-12-01
Saumye Singh	SBI Bank	2345678923456789	Debit	2026-09-15
Aarav Mehta	ICICI Bank	3456789034567890	Credit	2028-06-20
Priya Sharma	Axis Bank	4567890145678901	Debit	2025-11-30
Rohan Verma	Kotak Bank	5678901256789012	Credit	2029-05-10
Ananya Roy	PNB	6789012367890123	Debit	2024-08-21
Kunal Das	Union Bank	7890123478901234	Credit	2027-01-15
Shreya Iyer	Canara Bank	8901234589012345	Debit	2026-04-05
Rahul Nair	Bank of Baroda	9012345690123456	Credit	2028-03-18
Neha Kapoor	IndusInd Bank	0123456701234567	Debit	2025-10-27
Arjun Desai	Yes Bank	1111222233334444	Credit	2027-07-13

9. Get QR Sharing Records with Pass Details (INNER JOIN)

```
SELECT Q.QRShare_Id, Q.QRCode, Q.CreatedAt, P.Pass_Name
FROM QRSHARING Q
INNER JOIN PASSBANKACCOUNT P ON Q.Pass_Id = P.Pass_Id;
```

QRShare_Id	QRCode	CreatedAt	Pass_Name
1	QR123ABC	2025-03-06 19:02:48	Amazon Prime Membership
2	QR456DEF	2025-03-06 19:02:48	Concert Ticket
3	QR789GHI	2025-03-06 19:02:48	50% Off Coupon
4	QR101JKL	2025-03-06 19:02:48	Gym Membership
5	QR112MNO	2025-03-06 19:02:48	Netflix Subscription
6	QR134PQR	2025-03-06 19:02:48	Flight Ticket
7	QR156STU	2025-03-06 19:02:48	Amazon Gift Card
8	QR178VWX	2025-03-06 19:02:48	PlayStation Plus Membership
9	QR190YZA	2025-03-06 19:02:48	Movie Ticket
10	QR202BCD	2025-03-06 19:02:48	Tech Conference Entry
11	QR214EFG	2025-03-06 19:02:48	Udemy Subscription

10. Get Users and Their Multi-Device Access Records (FULL OUTER JOIN – Simulated)

```
SELECT U.User_Id, U.User_Name, M.Device_Id, M.Connected_Devices
FROM USERS U
LEFT JOIN MULTIDEVICEACCESS M ON U.User_Id = M.Access_Id
UNION
SELECT U.User_Id, U.User_Name, M.Device_Id, M.Connected_Devices
FROM USERS U
RIGHT JOIN MULTIDEVICEACCESS M ON U.User_Id = M.Access_Id;
```

User_Id	User_Name	Device_Id	Connected_Devices
1	Vineet Sahoo	Device001	Laptop, Mobile
2	Saumye Singh	Device002	Mobile
3	Aarav Mehta	Device003	Tablet, Mobile
4	Priya Sharma	Device004	Laptop, Tablet
5	Rohan Verma	Device005	Smartwatch, Mobile
6	Ananya Roy	Device006	Mobile, Smart TV
7	Kunal Das	Device007	Laptop, Mobile, Tablet
8	Shreya Iyer	Device008	Workstation, Mobile
9	Rahul Nair	Device009	Laptop, Home Assistant
10	Neha Kapoor	Device010	Smartphone, Wearable
11	Arjun Desai	Device011	Laptop, Smart Display

IV) GROUP BY QUERIES

1. Count the Number of Users per Email Domain

```
SELECT SUBSTRING_INDEX(Email, '@', -1) AS Domain, COUNT(*) AS UserCount
FROM USERS
GROUP BY Domain;
```

Domain	UserCount
example.com	11

2. Count the Number of Transactions per User

```
SELECT User_Id, COUNT(Transaction_Id) AS Total_Transactions
FROM TRANSACTIONS
GROUP BY User_Id;
```

User_Id	Total_Transactions
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

3. Count the Notifications Sent per User

```
SELECT User_Id, COUNT(Notification_Id) AS Total_Notifications
FROM NOTIFICATIONS
GROUP BY User_Id;
```

User_Id	Total_Notifications
1	1
2	2
3	1
4	2
5	1
6	2
7	1
8	2
9	1
10	2
11	1

4. Get the Latest Backup Date per User

```
SELECT User_Id, MAX(Backup_Date) AS Last_Backup_Date
FROM BACKUPRECOVERY
GROUP BY User_Id;
```

User_Id	Last_Backup_Date
1	2025-03-01 00:00:00
2	2025-03-02 00:00:00
3	2025-03-03 00:00:00
4	2025-03-04 00:00:00
5	2025-03-05 00:00:00
6	2025-03-06 00:00:00
7	2025-03-07 00:00:00
8	2025-03-08 00:00:00
9	2025-03-09 00:00:00
10	2025-03-10 00:00:00
11	2025-03-11 00:00:00

5. Find the Number of Cards per Bank Name

```
SELECT BankName, COUNT(Card_No) AS Total_Cards
FROM BANK
GROUP BY BankName;
```

BankName	Total_Cards
HDFC Bank	1
SBI Bank	1
ICICI Bank	1
Axis Bank	1
Kotak Bank	1
PNB	1
Union Bank	1
Canara Bank	1
Bank of Baroda	1
IndusInd Bank	1
Yes Bank	1

6. Calculate the Average Number of Connected Devices per Multi-Device Access Entry

```
SELECT Device_Id, AVG(Connected_Devices) AS Avg_Connected_Devices
FROM MULTIDEVICEACCESS
GROUP BY Device_Id;
```

Device_Id	Avg_Connected_Devices
Device001	0
Device002	0
Device003	0
Device004	0
Device005	0
Device006	0
Device007	0
Device008	0
Device009	0
Device010	0
Device011	0

7. Count the Number of Shared Passwords per User

```
SELECT SenderUser_Id, COUNT(Share_Id) AS Total_Shares
FROM PASSSHARING
GROUP BY SenderUser_Id;
```

SenderUser_Id	Total_Shares
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

8. Get the Most Recent Expiration Alert per User

```
SELECT User_Id, MAX(Expiration_Date) AS Last_Expiration_Alert
FROM EXPIRATIONALERTS
GROUP BY User_Id;
```

User_Id	Last_Expiration_Alert
1	2025-04-10
2	2025-03-15
3	2025-06-01
4	2025-07-20
5	2025-08-11
6	2025-09-05
7	2025-10-01
8	2025-11-12
9	2025-12-25
10	2026-01-30
11	2026-02-15

9. Count the Number of Unique Preferences per Category

```
SELECT Category, COUNT(DISTINCT User_Id) AS Users_Count
FROM USERPREFERENCES
GROUP BY Category;
```

Category	Users_Count
Cybersecurity	1
E-commerce	1
Education	1
Entertainment	1
Finance	1
Gaming	1
Health & Fitness	1
Investment	1
Productivity	1
Technology	1
Travel	1

10. Find the Oldest and Newest Password Created per User

SELECT User_Id, MIN(CreatedAt) AS Oldest_Password, MAX(CreatedAt) AS Newest_Password

FROM PASSWORDMANAGER

GROUP BY User_Id;

User_Id	Oldest_Password	Newest_Password
1	2025-03-06 18:12:22	2025-03-06 18:12:22
2	2025-03-06 18:12:22	2025-03-06 18:12:22
3	2025-03-06 18:12:22	2025-03-06 18:12:22
4	2025-03-06 18:12:22	2025-03-06 18:12:22
5	2025-03-06 18:12:22	2025-03-06 18:12:22
6	2025-03-06 18:12:22	2025-03-06 18:12:22
7	2025-03-06 18:12:22	2025-03-06 18:12:22
8	2025-03-06 18:12:22	2025-03-06 18:12:22
9	2025-03-06 18:12:22	2025-03-06 18:12:22
10	2025-03-06 18:12:22	2025-03-06 18:12:22
11	2025-03-06 18:12:22	2025-03-06 18:12:22

V) ORDER BY QUERIES

1. List All Users Sorted Alphabetically by Name

SELECT * FROM USERS

ORDER BY User_Name ASC;

User_Id	User_Name	Email	Password
3	Aarav Mehta	aarav@example.com	hashedpassword3
6	Ananya Roy	ananya@example.com	hashedpassword6
11	Arjun Desai	arjun@example.com	updatedpassword
7	Kunal Das	kunal@example.com	updatedpassword
10	Neha Kapoor	neha@example.com	hashedpassword10
4	Priya Sharma	priya@example.com	updatedpassword
9	Rahul Nair	rahul@example.com	hashedpassword9
5	Rohan Verma	rohan@example.com	hashedpassword5
2	Saumye Singh	saumye@example.com	hashedpassword2
8	Shreya Iyer	shreya@example.com	updatedpassword
1	Vineet Sahoo	vineet@example.com	updatedpassword

2. List Transactions Sorted by Most Recent First

SELECT * FROM TRANSACTIONS

ORDER BY Transaction_Date DESC;

Transaction_Id	User_Id	Transaction_Date	Transaction_Status
1	1	2025-03-06 19:01:38	Success
2	2	2025-03-06 19:01:38	Pending
3	3	2025-03-06 19:01:38	Failed
4	4	2025-03-06 19:01:38	Success
5	5	2025-03-06 19:01:38	Pending
6	6	2025-03-06 19:01:38	Failed
7	7	2025-03-06 19:01:38	Success
8	8	2025-03-06 19:01:38	Success
9	9	2025-03-06 19:01:38	Pending
10	10	2025-03-06 19:01:38	Failed
11	11	2025-03-06 19:01:38	Success

3. Display User Profiles Sorted by Phone Number

SELECT * FROM USERPROFILE

ORDER BY Phone_No ASC;

Profile_Id	User_Id	Phone_No	Bio	Profile_Picture
7	7	3210987654	Finance and investment expert	profile7.jpg
6	6	4321098765	Data science enthusiast	profile6.jpg
5	5	5432109876	Music lover and tech geek	profile5.jpg
11	11	6274930182	Cybersecurity and ethical hacking	profile11.jpg
4	4	6543210987	Love coding and open-source	profile4.jpg
10	10	7382910564	Full Stack Developer	profile10.jpg
3	3	7654321098	AI and ML enthusiast	profile3.jpg
9	9	8293745610	Passionate about UI/UX design	profile9.jpg
2	2	8765432109	Cybersecurity researcher	profile2.jpg
8	8	9102837465	Blockchain and Web3 Developer	profile8.jpg
1	1	9876543210	Tech enthusiast and developer	profile1.jpg

4. Show Notifications in Reverse Chronological Order

SELECT * FROM NOTIFICATIONS

ORDER BY Timestamp DESC;

Notification_Id	User_Id	Message	Read_Status	Timestamp
12	2	Your pass "Concert Ticket" has expired!	0	2025-03-29 13:04:25
13	4	Your pass "Gym Membership" has expired!	0	2025-03-29 13:04:25
14	6	Your pass "Flight Ticket" has expired!	0	2025-03-29 13:04:25
15	8	Your pass "PlayStation Plus Membership" has ex...	0	2025-03-29 13:04:25
16	10	Your pass "Tech Conference Entry" has expired!	0	2025-03-29 13:04:25
1	1	Your pass is expiring soon!	0	2025-03-06 19:01:49
2	2	New security update available.	1	2025-03-06 19:01:49
3	3	Pass shared successfully.	0	2025-03-06 19:01:49
4	4	Your transaction was successful.	1	2025-03-06 19:01:49
5	5	Backup completed successfully.	0	2025-03-06 19:01:49
6	6	New device login detected.	1	2025-03-06 19:01:49
7	7	Pass updated successfully.	0	2025-03-06 19:01:49
8	8	Upcoming event reminder!	0	2025-03-06 19:01:49
9	9	Discount coupon expires soon.	1	2025-03-06 19:01:49
10	10	Password changed successfully.	0	2025-03-06 19:01:49
11	11	Security alert: Unusual login detected.	1	2025-03-06 19:01:49

5. Show Expiring Alerts with Earliest Expiration First

SELECT * FROM EXPIRATIONALERTS

ORDER BY Expiration_Date ASC;

Alert_Id	Pass_Id	User_Id	Expiration_Date	Status
2	2	2	2025-03-15	Expired
1	1	1	2025-04-10	Active
3	3	3	2025-06-01	Active
4	4	4	2025-07-20	Expired
5	5	5	2025-08-11	Active
6	6	6	2025-09-05	Expired
7	7	7	2025-10-01	Active
8	8	8	2025-11-12	Expired
9	9	9	2025-12-25	Active
10	10	10	2026-01-30	Expired
11	11	11	2026-02-15	Active

6. Sort Bank Accounts by Expiry Date (Newest First)

SELECT * FROM BANK

ORDER BY Expiry_Date DESC;

Bank_Id	User_Id	BankName	Card_No	Card_Type	Expiry_Date
5	5	Kotak Bank	5678901256789012	Credit	2029-05-10
3	3	ICICI Bank	3456789034567890	Credit	2028-06-20
9	9	Bank of Baroda	9012345690123456	Credit	2028-03-18
1	1	HDFC Bank	1234567812345678	Credit	2027-12-01
11	11	Yes Bank	1111222233334444	Credit	2027-07-13
7	7	Union Bank	7890123478901234	Credit	2027-01-15
2	2	SBI Bank	2345678923456789	Debit	2026-09-15
8	8	Canara Bank	8901234589012345	Debit	2026-04-05
4	4	Axis Bank	4567890145678901	Debit	2025-11-30
10	10	IndusInd Bank	0123456701234567	Debit	2025-10-27
6	6	PNB	6789012367890123	Debit	2024-08-21

7. List Multi-Device Access Sorted by Number of Connected Devices

SELECT * FROM MULTIDEVICEACCESS

ORDER BY Connected_Devices DESC;

Access_Id	Device_Id	Last_IP	Connected_Devices
8	Device008	192.168.1.80	Workstation, Mobile
3	Device003	192.168.1.30	Tablet, Mobile
5	Device005	192.168.1.50	Smartwatch, Mobile
10	Device010	192.168.2.10	Smartphone, Wearable
6	Device006	192.168.1.60	Mobile, Smart TV
2	Device002	192.168.1.20	Mobile
4	Device004	192.168.1.40	Laptop, Tablet
11	Device011	192.168.2.20	Laptop, Smart Display
7	Device007	192.168.1.70	Laptop, Mobile, Tablet
1	Device001	192.168.1.10	Laptop, Mobile
9	Device009	192.168.1.90	Laptop, Home Assistant

8. Show Passwords Sorted by Their Creation Date (Oldest First)

```
SELECT * FROM PASSWORDMANAGER
ORDER BY CreatedAt ASC;
```

Password_Id	User_Id	Encrypted_Password	CreatedAt
1	1	encryptedpass1	2025-03-06 18:12:22
2	2	encryptedpass2	2025-03-06 18:12:22
3	3	encryptedpass3	2025-03-06 18:12:22
4	4	encryptedpass4	2025-03-06 18:12:22
5	5	encryptedpass5	2025-03-06 18:12:22
6	6	encryptedpass6	2025-03-06 18:12:22
7	7	encryptedpass7	2025-03-06 18:12:22
8	8	encryptedpass8	2025-03-06 18:12:22
9	9	encryptedpass9	2025-03-06 18:12:22
10	10	encryptedpass10	2025-03-06 18:12:22
11	11	encryptedpass11	2025-03-06 18:12:22

9. Show Pass Sharing Records Sorted by Shared Date (Most Recent First)

```
SELECT * FROM PASSSHARING
ORDER BY SharedAt DESC;
```

Share_Id	SenderUser_Id	ReceiverUser_Id	Pass_Id	SharedAt	Status
1	1	2	1	2025-03-06 18:14:08	Accepted
2	2	3	2	2025-03-06 18:14:08	Pending
3	3	4	3	2025-03-06 18:14:08	Rejected
4	4	5	4	2025-03-06 18:14:08	Accepted
5	5	6	5	2025-03-06 18:14:08	Pending
6	6	7	6	2025-03-06 18:14:08	Rejected
7	7	8	7	2025-03-06 18:14:08	Accepted
8	8	9	8	2025-03-06 18:14:08	Pending
9	9	10	9	2025-03-06 18:14:08	Accepted
10	10	11	10	2025-03-06 18:14:08	Rejected
11	11	1	11	2025-03-06 18:14:08	Accepted

10. Show User Preferences Sorted by Category

SELECT * FROM USERPREFERENCES

ORDER BY Category ASC;

Preference_Id	User_Id	Category
8	8	Cybersecurity
6	6	E-commerce
7	7	Education
5	5	Entertainment
2	2	Finance
9	9	Gaming
4	4	Health & Fitness
11	11	Investment
10	10	Productivity
1	1	Technology
3	3	Travel

VI) UNION, UNION ALL ETC QUERIES

1. UNION - Combine All User IDs from Users and Password Manager (Unique Results)

SELECT User_Id FROM USERS

UNION

SELECT User_Id FROM PASSWORDMANAGER;

User_Id
3
6
11
7
10
4
9
5
2
8
1

2. UNION ALL - Combine User Emails from Users and UserProfile (Includes Duplicates)

SELECT Email FROM USERS

UNION ALL

SELECT Email FROM USERPROFILE;

Tables_in_passvault
activefeatures
backuprecovery
backupstatusview
bank
dashboard
expirationalerts
expiringpasswords
features
multideviceaccess
multideviceaccessview
notifications
passbankaccount
passsharing
passsync
passwordmanager
passwordmanagerview
passwordsharingview
qrsharing
transactions
transactionsview
usernotifications
userpreferences
userpreferencesview
userprofile
userprofileview

3. UNION - Get All User IDs from UserProfile and Password Manager (Distinct)

```
SELECT User_Id FROM USERPROFILE
UNION
SELECT User_Id FROM PASSWORDMANAGER;
```

User_Id
1
2
3
4
5
6
7
8
9
10
11

4. UNION ALL - Get All Transactions and Bank Details (Including Duplicates)

```
SELECT User_Id, Transaction_Date FROM TRANSACTIONS
UNION ALL
SELECT User_Id, Expiry_Date FROM BANK;
```

User_Id	Transaction_Date
1	2025-03-06 19:01:38
2	2025-03-06 19:01:38
3	2025-03-06 19:01:38
4	2025-03-06 19:01:38
5	2025-03-06 19:01:38
6	2025-03-06 19:01:38
7	2025-03-06 19:01:38
8	2025-03-06 19:01:38
9	2025-03-06 19:01:38
10	2025-03-06 19:01:38
11	2025-03-06 19:01:38
1	2027-12-01 00:00:00
2	2026-09-15 00:00:00
3	2028-06-20 00:00:00
4	2025-11-30 00:00:00
5	2029-05-10 00:00:00
6	2024-08-21 00:00:00
7	2027-01-15 00:00:00
8	2026-04-05 00:00:00
9	2028-03-18 00:00:00
10	2025-10-27 00:00:00
11	2027-07-13 00:00:00

5. INTERSECT - Get Common User IDs Present in Users and Dashboard

```
SELECT User_Id FROM USERS
INTERSECT
SELECT User_Id FROM DASHBOARD;
```

User_Id
3
6
11
7
10
4
9
5
2
8
1

6. INTERSECT - Find Users Who Have Both Passwords and Backups

```
SELECT User_Id FROM PASSWORDMANAGER
INTERSECT
SELECT User_Id FROM BACKUPRECOVERY;
```

User_Id
1
2
3
4
5
6
7
8
9
10
11

7. INTERSECT - Find Users Who Have Multi-Device Access and User Preferences

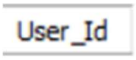
```
SELECT User_Id
FROM USERPREFERENCES u
WHERE EXISTS (
   SELECT 1 FROM MULTIDEVICEACCESS m WHERE u.User_Id = m.Access_Id
);
```

User_Id
1
2
3
4
5
6
7
8
9
10
11

8. MINUS - Find Users Who Have a Profile but No Password Stored

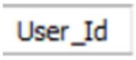
```
SELECT User_Id
```

```
FROM USERPROFILE
WHERE User_Id NOT IN (SELECT User_Id FROM PASSWORDMANAGER);
```



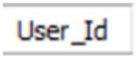
9. MINUS - Find Transactions That Do Not Have a Matching Bank Record

```
SELECT t.User_Id
FROM TRANSACTIONS t
LEFT JOIN BANK b ON t.User_Id = b.User_Id
WHERE b.User_Id IS NULL;
```



10. MINUS - Find Users Who Have Bank Accounts but No Notifications

```
SELECT User_Id
FROM BANK b
WHERE NOT EXISTS (
   SELECT 1 FROM NOTIFICATIONS n WHERE b.User_Id = n.User_Id
);
```



VII) HAVING CLUASE

1. Find Users Who Have More Than 0 Bank Account

```
SELECT User_Id, COUNT(Bank_Id) AS Total_Accounts
FROM BANK
GROUP BY User_Id
HAVING COUNT(Bank_Id) > 0;
```

User_Id	Total_Accounts
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

2. Find Users Who Have Made More Than 0 Transactions

```
SELECT User_Id, COUNT(Transaction_Id) AS Total_Transactions
FROM TRANSACTIONS
GROUP BY User_Id
HAVING COUNT(Transaction_Id) > 0;
```

User_Id	Total_Transactions
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

3. Find Users Who Have exactly 1 Preferences

```
SELECT User_Id, COUNT(Preference_Id) AS Total_Preferences
```

```
FROM USERPREFERENCES
GROUP BY User_Id
HAVING COUNT(Preference_Id) =1;
```

User_Id	Total_Preferences
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

4. Find Users Who Have Received More Than 1 Notifications

```
SELECT User_Id, COUNT(Notification_Id) AS Notification_Count
FROM NOTIFICATIONS
GROUP BY User_Id
HAVING COUNT(Notification_Id) > 1;
```

User_Id	Notification_Count
2	2
4	2
6	2
8	2
10	2

5. Find Users Who Have More Than 1 Multi-Device Access Record

```
SELECT User_Id, COUNT(Access_Id) AS Access_Count
FROM MULTIDEVICEACCESS
GROUP BY User_Id
HAVING COUNT(Access_Id) > 1;
```

Tables_in_passvault
activefeatures
backuprecovery
backupstatusview
bank
dashboard
expirationalerts
expiringpasswords
features
multideviceaccess
multideviceaccessview
notifications
passbankaccount
passsharing
passsync
passwordmanager
passwordmanagerview
passwordsharingview
qrsharing
transactions
transactionsview
usernotifications
userpreferences
userpreferencesview
userprofile
userprofileview

6. Find Users Who Have Exactly 1 Expired Passes

```
SELECT User_Id, COUNT(Alert_Id) AS Expired_Passes
FROM EXPIRATIONALERTS
WHERE Status = 'Expired'
GROUP BY User_Id
HAVING COUNT(Alert_Id) = 1;
```

User_Id	Expired_Passes
2	1
4	1
6	1
8	1
10	1

7. Find Users Who Have More Than 0 QR Codes Shared

```
SELECT User_Id, COUNT(QRShare_Id) AS QR_Shares
FROM QRSHARING
GROUP BY User_Id
HAVING COUNT(QRShare_Id) > 0;
```

User_Id	QR_Shares
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

8. Find Users Who Have More Than 0 Password Backup

```
SELECT User_Id, COUNT(Backup_Id) AS Backup_Count
FROM BACKUPRECOVERY
GROUP BY User_Id
HAVING COUNT(Backup_Id) > 0;
```

User_Id	Backup_Count
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

9. Find Users Who Have More Than 0 Passwords Stored

```
SELECT User_Id, COUNT(Password_Id) AS Total_Passwords
FROM PASSWORDMANAGER
GROUP BY User_Id
HAVING COUNT(Password_Id) > 0;
```

User_Id	Total_Passwords
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

10. Find Users Who Have Shared More Than 3 Passes

```
SELECT SenderUser_Id, COUNT(Share_Id) AS Total_Shares
FROM PASSSHARING
GROUP BY SenderUser_Id
HAVING COUNT(Share_Id) > 3;
```

SenderUser_Id	Total_Shares
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1

VIII) SUBQUERIES, NESTED SUBQUERIES & COORELATED SUBQUERIES

1. Subquery – Find Users Who Have Made Transactions

```
SELECT User_Id, User_Name
FROM USERS
WHERE User_Id IN (SELECT DISTINCT User_Id FROM TRANSACTIONS);
```

User_Id	User_Name
1	Vineet Sahoo
2	Saumye Singh
3	Aarav Mehta
4	Priya Sharma
5	Rohan Verma
6	Ananya Roy
7	Kunal Das
8	Shreya Iyer
9	Rahul Nair
10	Neha Kapoor
11	Arjun Desai

2. Nested Subquery – Find Users Who Have the Most Transactions

```
SELECT User_Id, User_Name
FROM USERS
WHERE User_Id = (SELECT User_Id FROM TRANSACTIONS
                 GROUP BY User_Id
```

ORDER BY COUNT(Transaction_Id) DESC LIMIT 1);

User_Id	User_Name
1	Vineet Sahoo

3. Correlated Subquery – Find Users Who Have More Than 0 Bank Accounts

SELECT User_Id, User_Name

FROM USERS U

WHERE (SELECT COUNT(*) FROM BANK B WHERE B.User_Id = U.User_Id) > 0;

User_Id	User_Name
1	Vineet Sahoo
2	Saumye Singh
3	Aarav Mehta
4	Priya Sharma
5	Rohan Verma
6	Ananya Roy
7	Kunal Das
8	Shreya Iyer
9	Rahul Nair
10	Neha Kapoor
11	Arjun Desai

4. Subquery – Find Users Who Have Not Set Any Preferences

SELECT User_Id, User_Name

FROM USERS

WHERE User_Id IN (SELECT DISTINCT User_Id FROM USERPREFERENCES);

User_Id	User_Name
NULL	NULL

5. Nested Subquery – Find Users Who Have the Oldest Password Backup

SELECT User_Id, User_Name

FROM USERS

WHERE User_Id = (SELECT User_Id FROM BACKUPRECOVERY

ORDER BY Backup_Date ASC LIMIT 1);

User_Id	User_Name
1	Vineet Sahoo
NULL	NULL

6. Correlated Subquery – Find Users Who Have More Notifications Than the Average

SELECT User_Id, User_Name

FROM USERS U

WHERE (SELECT COUNT(*) FROM NOTIFICATIONS N WHERE N.User_Id = U.User_Id)

> (SELECT AVG(Notification_Count) FROM

(SELECT COUNT(*) AS Notification_Count FROM NOTIFICATIONS GROUP BY User_Id) AS AvgNotifications);

User_Id	User_Name
2	Saumye Singh
4	Priya Sharma
6	Ananya Roy
8	Shreya Iyer
10	Neha Kapoor
NULL	NULL

7. Subquery – Find Users Who Have Shared More Than 0 QR Codes

SELECT User_Id, User_Name

FROM USERS

WHERE User_Id IN (SELECT User_Id FROM QRSHARING

GROUP BY User_Id

HAVING COUNT(QRShare_Id) > 0);

User_Id	User_Name
1	Vineet Sahoo
2	Saumye Singh
3	Aarav Mehta
4	Priya Sharma
5	Rohan Verma
6	Ananya Roy
7	Kunal Das
8	Shreya Iyer
9	Rahul Nair
10	Neha Kapoor
11	Arjun Desai

8. Nested Subquery – Find Users Who Have the Most Shared Passwords

```
SELECT User_Id, User_Name
FROM USERS
WHERE User_Id = (SELECT SenderUser_Id FROM PASSSHARING
        GROUP BY SenderUser_Id
        ORDER BY COUNT(Share_Id) DESC LIMIT 1);
```

User_Id	User_Name
1	Vineet Sahoo
NULL	NULL

9. Correlated Subquery – Find Users Who Have Received More Notifications Than They Have Transactions

```
SELECT User_Id, User_Name
FROM USERS U
WHERE (SELECT COUNT(*) FROM NOTIFICATIONS N WHERE N.User_Id =
U.User_Id)
    > (SELECT COUNT(*) FROM TRANSACTIONS T WHERE T.User_Id =
U.User_Id);
```

User_Id	User_Name
2	Saumye Singh
4	Priya Sharma
6	Ananya Roy
8	Shreya Iyer
10	Neha Kapoor

10. Subquery – Find Users Who Have No Transactions

```
SELECT User_Id, User_Name
FROM USERS
WHERE User_Id NOT IN (SELECT DISTINCT User_Id FROM TRANSACTIONS);
```

User_Id	User_Name
NULL	NULL

IX) TRIGGER FUNCTION

1. Trigger to Log Every New Transaction

```
DELIMITER $$

CREATE TRIGGER log_transaction
AFTER INSERT ON TRANSACTIONS
FOR EACH ROW
BEGIN
    INSERT INTO NOTIFICATIONS (User_Id, Message, Read_Status, Timestamp)
    VALUES (NEW.User_Id, CONCAT('New transaction recorded: ', NEW.Transaction_Id),
0, NOW());
END$$

DELIMITER ;

SELECT * FROM NOTIFICATIONS WHERE User_Id = 1 ORDER BY Timestamp
```

DESC;

Notification_Id	User_Id	Message	Read_Status	Timestamp
37	1	New transaction recorded: 12	0	2025-04-01 07:12:09
1	1	Your pass is expiring soon!	0	2025-03-06 19:01:49

2. Trigger to Update Backup Status Automatically

```
DELIMITER $$

CREATE TRIGGER update_backup_status
BEFORE INSERT ON BACKUPRECOVERY
FOR EACH ROW
BEGIN
   SET NEW.Recovery_Status = 'Pending';
END$$

DELIMITER ;
```

SELECT * FROM BACKUPRECOVERY WHERE Backup_Id = 12;

Backup_Id	User_Id	Backup_Date	Recovery_Status	Backup_Location
12	2	2025-04-01 07:54:38	Pending	Cloud Storage

3. Trigger to Prevent Duplicate QR Codes

```
DELIMITER $$

CREATE TRIGGER prevent_duplicate_qrcode
BEFORE INSERT ON QRSHARING
FOR EACH ROW
BEGIN
   DECLARE duplicate_count INT DEFAULT 0;

   -- Check if the QRCode already exists
   SELECT COUNT(*) INTO duplicate_count
```

```
    FROM QRSHARING
    WHERE QRCode = NEW.QRCode;

    -- If a duplicate is found, raise an error
    IF duplicate_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Duplicate QR Code is not allowed!';
    END IF;
END $$

DELIMITER ;

INSERT INTO QRSHARING (Pass_Id,User_Id, QRCode, CreatedAt, IsScanned)
VALUES (12,12, 'QR123ABC', NOW(), 0);
```

Error Code: 1644. Duplicate QR Code is not allowed!

4. Trigger to Record Last Login for Users

```
DELIMITER $$

CREATE TRIGGER update_last_login
AFTER UPDATE ON USERS
FOR EACH ROW
BEGIN
    -- Update the Last_Login field in the DASHBOARD table
    UPDATE DASHBOARD
    SET Last_Login = NOW()
    WHERE Dashboard_Id = NEW.User_Id;
END $$

DELIMITER ;

ALTER TABLE DASHBOARD ADD COLUMN Last_Login DATETIME;
```

UPDATE USERS SET Password = 'newhashedpassword' WHERE User_Id = 5;

SELECT * FROM DASHBOARD;

Dashboard_Id	User_Id	Last_Login
1	1	NULL
2	2	NULL
3	3	NULL
4	4	NULL
5	5	2025-04-02 11:57:02
6	6	NULL
7	7	NULL
8	8	NULL
9	9	NULL
10	10	NULL
11	11	NULL
NULL	NULL	NULL

5. Trigger to Auto-Delete Expired Passwords

```
DELIMITER $$

CREATE TRIGGER delete_expired_passwords
AFTER INSERT ON PASSWORDMANAGER
FOR EACH ROW
BEGIN
   DELETE FROM PASSWORDMANAGER
   WHERE CreatedAt < DATE_SUB(NOW(), INTERVAL 1 YEAR);
END

$$DELIMITER ;
```

SELECT * FROM PASSWORDMANAGER WHERE CreatedAt < DATE_SUB(NOW(), INTERVAL 1 YEAR);

Password_Id	User_Id	Encrypted_Password	CreatedAt
12	1	oldpassword1	2023-03-01 10:00:00
13	2	oldpassword2	2023-02-15 12:30:00

6. Trigger to Ensure Unique Bank Card Numbers

```
DELIMITER $$

CREATE TRIGGER prevent_duplicate_card
BEFORE INSERT ON BANK
FOR EACH ROW
BEGIN
   IF EXISTS (SELECT 1 FROM BANK WHERE Card_No = NEW.Card_No) THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Card number already exists!';
   END IF;
END $$

DELIMITER ;

INSERT INTO BANK (Bank_Id, User_Id, BankName, Card_No, Card_Type,
Expiry_Date)
VALUES (12, 2, 'Federal Bank', '9999888877776666', 'Credit', '2029-12-31');

INSERT INTO BANK (Bank_Id, User_Id, BankName, Card_No, Card_Type,
Expiry_Date)
VALUES (13, 3, 'HDFC Bank', '1234567812345678', 'Credit', '2028-01-01');
```

Error Code: 1644. Card number already exists!

7. Trigger to Send Notification for Expiring Passwords

```
DELIMITER $$

CREATE TRIGGER before_insert_expired_id
BEFORE INSERT ON expirationalerts
FOR EACH ROW
BEGIN
   IF NEW.expiration_date BETWEEN CURDATE() AND DATE_ADD(CURDATE(),
INTERVAL 7 DAY) THEN
```

```
    INSERT INTO notifications (user_id, message, timestamp)
    VALUES (NEW.user_id, 'Your ID is about to expire soon!', NOW());
  END IF;

     IF NEW.expiration_date < CURDATE() THEN
    INSERT INTO notifications (user_id, message, timestamp)
    VALUES (NEW.user_id, 'Your ID has expired!', NOW());
  END IF;
END $$

DELIMITER ;

select * from notifications
where message ="Your ID has expired!" OR message ="Your ID is about to expire soon!"
order by timestamp desc;
```

Notification_Id	User_Id	Message	Read_Status	Timestamp
63	4	Your ID is about to expire soon!	0	2025-04-01 10:20:20
64	5	Your ID has expired!	0	2025-04-01 10:20:20
62	3	Your ID is about to expire soon!	0	2025-04-01 10:13:31
61	2	Your ID has expired!	0	2025-04-01 10:08:12
NULL	NULL	NULL	NULL	NULL

8. Trigger to Validate Password Length

```
CREATE TRIGGER validate_password_length
BEFORE INSERT ON PASSWORDMANAGER
FOR EACH ROW
BEGIN
  IF CHAR_LENGTH(NEW.Encrypted_Password) < 8 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Password must be at least 8 characters long!';
  END IF;
END $$
```

```
DELIMITER ;

INSERT INTO PASSWORDMANAGER (Password_Id, User_Id, Encrypted_Password, CreatedAt)
VALUES (12, 5, 'short1', NOW());

Error Code: 1644. Password must be at least 8 characters long!

INSERT INTO PASSWORDMANAGER (Password_Id, User_Id, Encrypted_Password, CreatedAt)
VALUES (13, 6, 'SecurePass123', NOW());

Inserted successfully!!
```

9. Trigger to Delete User Data When Account is Removed

```
DELIMITER $$

CREATE TRIGGER delete_user_data
AFTER DELETE ON USERS
FOR EACH ROW
BEGIN
   DELETE FROM USERPROFILE WHERE User_Id = OLD.User_Id;
   DELETE FROM BANK WHERE User_Id = OLD.User_Id;
   DELETE FROM TRANSACTIONS WHERE User_Id = OLD.User_Id;
   DELETE FROM NOTIFICATIONS WHERE User_Id = OLD.User_Id;
END $$

DELIMITER ;

SELECT * FROM USERS WHERE User_Id = 3;
SELECT * FROM USERPROFILE WHERE User_Id = 3;
SELECT * FROM BANK WHERE User_Id = 3;
```

SELECT * FROM TRANSACTIONS WHERE User_Id = 3;

SELECT * FROM NOTIFICATIONS WHERE User_Id = 3;

User_Id	User_Name	Email	Password
3	Aarav Mehta	aarav@example.com	hashedpassword3

Bank_Id	User_Id	BankName	Card_No	Card_Type	Expiry_Date
3	3	ICICI Bank	3456789034567890	Credit	2028-06-20

Notification_Id	User_Id	Message	Read_Status	Timestamp
3	3	Welcome back, Aarav!	0	2025-04-07 19:27:53

DELETE FROM USERS WHERE User_Id = 3;

SELECT * FROM USERS WHERE User_Id = 3;

SELECT * FROM USERPROFILE WHERE User_Id = 3;

SELECT * FROM BANK WHERE User_Id = 3;

SELECT * FROM TRANSACTIONS WHERE User_Id = 3;

SELECT * FROM NOTIFICATIONS WHERE User_Id = 3;

User_Id	User_Name	Email	Password
NULL	NULL	NULL	NULL

Profile_Id	User_Id	Phone_No	Bio	Profile_Picture
NULL	NULL	NULL	NULL	NULL

Bank_Id	User_Id	BankName	Card_No	Card_Type	Expiry_Date
NULL	NULL	NULL	NULL	NULL	NULL

Transaction_Id	User_Id	Transaction_Date	Transaction_Status
NULL	NULL	NULL	NULL

Notification_Id	User_Id	Message	Read_Status	Timestamp
NULL	NULL	NULL	NULL	NULL

10. Trigger to Auto-Update Profile When a New Phone Number is Added

DELIMITER $$

CREATE TRIGGER update_profile_phone

AFTER UPDATE ON USERPROFILE

FOR EACH ROW

```
BEGIN
    UPDATE USERS
    SET Email = CONCAT(NEW.Phone_No, '@user.com')
    WHERE User_Id = NEW.User_Id;
END $$

DELIMITER ;

SELECT * FROM USERS WHERE User_Id = 1;
```

User_Id	User_Name	Email	Password
1	Vineet Sahoo	vineet@example.com	updatedpassword

```
UPDATE USERPROFILE
SET Phone_No = '9998887776'
WHERE User_Id = 1;

SELECT * FROM USERS WHERE User_Id = 1;
```

User_Id	User_Name	Email	Password
1	Vineet Sahoo	9998887776@user.com	updatedpassword

Showing all created triggers:

```
SHOW TRIGGERS FROM Passvault;
```

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer
update_backup_status	INSERT	backuprecovery	SET NEW.Recovery_Status = 'Pending'	BEFORE	2025-03-29 14:55:04.75	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
prevent_duplicate_card	INSERT	bank	BEGIN IF EXISTS (SELECT 1 FROM BANK WH...	BEFORE	2025-03-29 14:57:12.53	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
notify_expiring_passwords	INSERT	expirationalerts	INSERT INTO NOTIFICATIONS (User_Id, Messa...	AFTER	2025-03-29 14:57:22.87	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
delete_expired_passwords	INSERT	passwordmanager	DELETE FROM PASSWORDMANAGER WHERE Cr...	BEFORE	2025-03-29 14:56:57.07	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
validate_password_length	INSERT	passwordmanager	BEGIN IF LENGTH(NEW.Encrypted_Password...	BEFORE	2025-03-29 14:57:30.68	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
prevent_duplicate_qrcode	INSERT	qrsharing	BEGIN DECLARE duplicate_count INT; -...	BEFORE	2025-03-29 14:56:11.18	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
log_transaction	INSERT	transactions	INSERT INTO NOTIFICATIONS (User_Id, Messa...	AFTER	2025-03-29 14:54:31.80	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
update_profile_phone	UPDATE	userprofile	UPDATE USERS SET Email = CONCAT(NEW.Pho...	AFTER	2025-03-29 14:57:49.23	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
update_last_login	UPDATE	users	UPDATE DASHBOARD SET Dashboard_Id = NE...	AFTER	2025-03-29 14:56:48.24	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost
delete_user_data	DELETE	users	BEGIN DELETE FROM USERPROFILE WHERE ...	AFTER	2025-03-29 14:57:39.32	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost