SQUAD 4

HACKATHON-2 (pnb-28)

TASK 1: R. Sruthi

Design `reminder_settings` table: `reminder_id`, `user_id`, `days_before_due`.

Code :

```
CREATE TABLE ReminderSettings (

    reminder_id VARCHAR2(50) PRIMARY KEY,

    user_id VARCHAR2(50),

    days_before_due NUMBER,

    reminder_frequency VARCHAR2(50),

    reminder_start_date DATE,

    custom_message VARCHAR2(255),

    notification_preference VARCHAR2(50),

    bill_id VARCHAR2(50),

    FOREIGN KEY (bill_id) REFERENCES Bills(bill_id),

    FOREIGN KEY (user_id) REFERENCES Users(user_id)

);
```

Output Screenshot :

```
SQL> CREATE TABLE ReminderSettings (
  2      reminder_id VARCHAR2(50) PRIMARY KEY,
  3      user_id VARCHAR2(50),
  4      days_before_due NUMBER,
  5      reminder_frequency VARCHAR2(50),
  6      reminder_start_date DATE,
  7      custom_message VARCHAR2(255),
  8      notification_preference VARCHAR2(50),
  9      bill_id VARCHAR2(50),
 10      FOREIGN KEY (bill_id) REFERENCES Bills(bill_id),
 11      FOREIGN KEY (user_id) REFERENCES Users(user_id)
 12  );

Table created.
```

TASK 2: Sunil Kumar

Add `CHECK (days_before_due BETWEEN 1 AND 10)`

Code:

```
ALTER TABLE ReminderSettings

ADD CONSTRAINT chk_days_before_due

CHECK (days_before_due BETWEEN 1 AND 10);
```

Output:

```
SQL> CREATE TABLE ReminderSettings (
  2      reminder_id VARCHAR2(50) PRIMARY KEY,
  3      user_id VARCHAR2(50),
  4      days_before_due NUMBER,
  5      reminder_frequency VARCHAR2(50),
  6      reminder_start_date DATE,
  7      custom_message VARCHAR2(255),
  8      notification_preference VARCHAR2(50),
  9      bill_id VARCHAR2(50));

Table created.

SQL> ALTER TABLE ReminderSettings
  2  ADD CONSTRAINT chk_days_before_due
  3  CHECK (days_before_due BETWEEN 1 AND 10);

Table altered.

SQL> |
```

TASK 3: Yajat Gupta

Insert reminders for at least 3 users and write a query to show who should be reminded today.

Query/Code:

-- Insert sample bills for users (assuming all due in August 2025)

```
INSERT INTO Bills VALUES ('B001', 'Electricity Bill', 'Utilities',
TO_DATE('2025-08-05','YYYY-MM-DD'), 1200, 'Monthly', NULL, NULL, 1, 0, NULL, 'U001');

INSERT INTO Bills VALUES ('B002', 'Water Bill', 'Utilities',
TO_DATE('2025-08-07','YYYY-MM-DD'), 800, 'Monthly', NULL, NULL, 1, 0, NULL, 'U002');

INSERT INTO Bills VALUES ('B003', 'Gas Bill', 'Utilities',
TO_DATE('2025-08-03','YYYY-MM-DD'), 600, 'Monthly', NULL, NULL, 1, 0, NULL, 'U003');



-- Insert reminders: user gets reminded a fixed number of days before due date

INSERT INTO ReminderSettings VALUES ('R001', 'U001', 4, 'Once',
TO_DATE('2025-07-25','YYYY-MM-DD'), 'Pay your bill soon!', 'Email', 'B001');

INSERT INTO ReminderSettings VALUES ('R002', 'U002', 6, 'Once',
TO_DATE('2025-07-30','YYYY-MM-DD'), 'Water bill due!', 'SMS', 'B002');

INSERT INTO ReminderSettings VALUES ('R003', 'U003', 2, 'Once',
TO_DATE('2025-07-20','YYYY-MM-DD'), 'Remember gas bill!', 'App', 'B003');

 -- to give reminders today

SELECT

    u.user_id,

    u.name,

    u.email,

    r.reminder_id,

    b.bill_id,

    b.bill_name,

    b.due_date,

    r.days_before_due,

    TO_CHAR(b.due_date - r.days_before_due, 'YYYY-MM-DD') AS reminder_date,

    r.notification_preference

FROM
```

```
    ReminderSettings r
    JOIN Bills b ON r.bill_id = b.bill_id
    JOIN Users u ON r.user_id = u.user_id
WHERE
    TO_DATE('2025-08-01', 'YYYY-MM-DD') = (b.due_date - r.days_before_due)
    AND r.reminder_start_date <= TO_DATE('2025-08-01', 'YYYY-MM-DD');
```

TASK 4: Sujay R

Create a procedure to print all users who need reminders for bills due in 3 days.

Code:

```
CREATE OR REPLACE PROCEDURE Send_Reminders_For_3_Days_Left AS
BEGIN
    FOR rec IN (
        SELECT
            u.user_id,
            u.name,
            u.email,
            b.bill_name,
            b.due_date,
            rs.custom_message,
            rs.notification_preference
        FROM
            ReminderSettings rs
        JOIN
            Bills b ON rs.bill_id = b.bill_id
```

```
        JOIN
            Users u ON rs.user_id = u.user_id
        WHERE
            rs.days_before_due = 3
            AND TRUNC(b.due_date) - rs.days_before_due = TRUNC(SYSDATE)
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Reminder for User: ' || rec.name || ' Email: ' || rec.email);
        DBMS_OUTPUT.PUT_LINE('Bill: ' || rec.bill_name || ' Due Date: ' ||
TO_CHAR(rec.due_date, 'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Message: ' || rec.custom_message);
        DBMS_OUTPUT.PUT_LINE('Notification Method: ' || rec.notification_preference);
        DBMS_OUTPUT.PUT_LINE('----------------------------------------');
    END LOOP;
END;
/
```

Output Screenshot:

```
SQL> CREATE OR REPLACE PROCEDURE Send_Reminders_For_3_Days_Left AS
  2  BEGIN
  3      FOR rec IN (
  4          SELECT
  5              u.user_id,
  6              u.name,
  7              u.email,
  8              b.bill_name,
  9              b.due_date,
 10              rs.custom_message,
 11              rs.notification_preference
 12          FROM
 13              ReminderSettings rs
 14          JOIN
 15              Bills b ON rs.bill_id = b.bill_id
 16          JOIN
 17              Users u ON rs.user_id = u.user_id
 18          WHERE
 19              rs.days_before_due = 3
 20              AND TRUNC(b.due_date) - rs.days_before_due = TRUNC(SYSDATE)
 21      ) LOOP
 22          DBMS_OUTPUT.PUT_LINE('Reminder for User: ' || rec.name || ' Email: ' || rec.email);
 23          DBMS_OUTPUT.PUT_LINE('Bill: ' || rec.bill_name || ' Due Date: ' || TO_CHAR(rec.due_date
, 'YYYY-MM-DD'));
 24          DBMS_OUTPUT.PUT_LINE('Message: ' || rec.custom_message);
 25          DBMS_OUTPUT.PUT_LINE('Notification Method: ' || rec.notification_preference);
 26          DBMS_OUTPUT.PUT_LINE('----------------------------------------');
 27      END LOOP;
 28  END;
 29  /

Procedure created.
```

```
SQL> BEGIN
  2      Send_Reminders_For_3_Days_Left;
  3  END;
  4  /
Reminder for User: User One Email: user1@example.com
Bill: Electricity Bill Due Date: 2025-08-04
Message: Pay Electricity Bill
Notification Method: Email
--------------------------------------------
Reminder for User: User One Email: user1@example.com
Bill: Mobile Recharge Due Date: 2025-08-04
Message: Recharge Mobile Today
Notification Method: SMS
--------------------------------------------
Reminder for User: User Two Email: user2@example.com
Bill: Gas Bill Due Date: 2025-08-04
Message: Pay Gas Bill Soon
Notification Method: Email
--------------------------------------------

PL/SQL procedure successfully completed.
```

TASK 5: Saloni Chaudhary (pnb-29)

Write a nested query to show users who have never paid any bill.

Code :

```
SELECT name, email

FROM users

WHERE user_id NOT IN (

   SELECT DISTINCT user_id

   FROM bills

   WHERE is_paid = 1

);
```

Output Screenshot:

```
NAME
--------------------------------------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
Priya Sharma
priya.s@example.com

Amit Singh
amit.singh@example.com

Sunita Rao
sunita.r@example.com


NAME
--------------------------------------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
Rajesh Verma
rajesh.v@example.com


SQL>
```

TASK 6 (Optional): Shivam Kurda (pnb-29)

Design an index on `bill.due_date` and explain when it helps performance.

Code with explanation and output :

```sql
CREATE INDEX due_date_idx ON bill(due_date);


-- HOW AN INDEX ON DUE_DATE OPTIMIZES PERFORMANCE --
-- 1. Enables faster lookups based on due_date. For example, when fetching records
--    from the past few days to send reminders, the index allows quick access without scanning the entire table.
-- 2. Improves the efficiency of range queries, making it faster to retrieve records
--    where due_date falls within a specified range.
-- 3. Speeds up queries that sort results by due_date, as the index maintains the column values
--    in sorted order, reducing or eliminating the need for an additional sort operation.
```

PROBLEMS 4 TERMINAL PORTS SCRIPT OUTPUT SQL HISTORY TASK MONITOR

```
Index DUE_DATE_IDX created.
```

HACKATHON-1

TASK 1: Saloni Chaudhary

Use `Comparator` interface to sort all bills by due date.

Code :

BillSorter.java

```
public class BillSorter {


    public static List<Bill> SortedBillsByDueDate(List<Bill> bills) {

        List<Bill> sortedBills = new ArrayList<>(bills);

        sortedBills.sort(Comparator.comparing(Bill::getDueDate));

        return sortedBills;

    }
```

TASK 2:

Group payments by mode (e.g., Credit Card, Net Banking, Cash On Delivery) using: `Collectors.groupingBy`

Code :

PaymentService.java:

```
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
public class PaymentService {

        private static PaymentService inst=null;

        public static PaymentService getInstance() {
                if(inst  == null) {
                        inst=new PaymentService();
                }
                return inst;
        }
        public  Map<String,Long> getModeCount(List<Payment> payments){

                Map<String, Long>
modeCount=payments.stream().collect(Collectors.groupingBy(Payment::getMode,Collectors.
counting()));
```

```
            return modeCount;
        }
}
```

Main.java:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.time.LocalDate;
public class Main {
   public static void main(String[] args) {
       List<Payment> payments=new ArrayList<>();

       payments.add(new Payment(101, 501, 1200.50, LocalDate.of(2025, 7, 20), "Credit
Card"));
       payments.add(new Payment(102, 502, 875.00, LocalDate.of(2025, 7, 21), "Credit
Card"));
       payments.add(new Payment(103, 503, 1500.75, LocalDate.of(2025, 7, 22), "Net
Banking"));
       payments.add(new Payment(104, 504, 630.00, LocalDate.of(2025, 7, 23), "Cash"));
       payments.add(new Payment(105, 505, 999.99, LocalDate.of(2025, 7, 24), "Debit
Card"));

       PaymentService pservice=PaymentService.getInstance();
       Map<String,Long> mp=pservice.getModeCount(payments);
       for(String key:mp.keySet()) {
             System.out.println("Number of payments using "+key+" are "+mp.get(key));
       }

   }
}
```

Output :

```
Problems  Console  Javadoc  Declaration  Terminal  Coverage  Servers
<terminated> Main (1) [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_21.0.7.v20250502-0916/jre/bin/java (01-Aug-2025, 4:20:47 pm – 4:20:49 pm e
Number of payments using Credit Card are 2
Number of payments using Net Banking are 1
Number of payments using Debit Card are 1
Number of payments using Cash are 1
```

TASK 3:

Create a utility class `DateUtil` to parse and format dates (`dd-MM-yyyy`)

Code:

```
package com.paypilot.util;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.time.format.DateTimeParseException;

public class DateUtil {

    // Formatter to handle "dd-MM-yyyy" date format

    private static final DateTimeFormatter FORMATTER =
DateTimeFormatter.ofPattern("dd-MM-yyyy");

    // Converts a string like "28-07-2025" to LocalDate

    public static LocalDate parse(String dateStr) {

        if (dateStr == null) {

            throw new IllegalArgumentException("Input date string cannot be null");

        }

        try {

            return LocalDate.parse(dateStr, FORMATTER);

        } catch (DateTimeParseException e) {

            throw new IllegalArgumentException("Invalid date format. Expected dd-MM-yyyy", e);

        }

    }

    // Converts a LocalDate to a string like "28-07-2025"

    public static String format(LocalDate date) {

        if (date == null) {

            throw new IllegalArgumentException("Input date cannot be null");

       }
```

```
        return date.format(FORMATTER);
    }
}
```

TASK 4:

Create a mock data generator class that returns 5 sample users, 10 bills, 5 payments

Code:

```
package com.paypilot.util;


import com.paypilot.model.Bill;
import com.paypilot.model.Payment;
import com.paypilot.model.User;


import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;


public class MockDataGenerator {
    private static final String[] BILL_NAMES = {
        "Water Bill", "Internet Bill", "Mobile Bill", "Gas Bill", "Electricity Bill"
    };


    private static final String[] CATEGORIES = {
        "Water", "Internet", "Mobile", "Gas", "Electricity"
```

```
};

private static final String[] PAYMENT_MODES = {
    "Credit Card", "NetBanking", "Cash on Delivery"
};

public static List<User> generateUsers() {
    List<User> users = new ArrayList<>();
    users.add(new User(1, "Hari", "hari@gmail.com", "9876543210", "pass123"));
    users.add(new User(2, "Vijay", "vijay@gmail.com", "8765432109", "bpass2"));
    users.add(new User(3, "Chandru", "chandru@gmail.com", "7654321098", "chaie1"));
    users.add(new User(4, "Neha", "neha@gmail.com", "6543210987", "huina123"));
    users.add(new User(5, "Cheera", "cheera@gmail.com", "8432109876", "huinn@123"));
    System.out.println("User Data Generated");
    return users;
}

public static List<Bill> generateBills() {
    List<Bill> bills = new ArrayList<>();
    for (int i = 1; i <= 10; i++) {
        int userId = (i % 5) + 1;
        int index = (i - 1) % BILL_NAMES.length;
        LocalDate dueDate = LocalDate.now().minusDays(i);
        bills.add(new Bill(
```

```
            userId,
            i,
            BILL_NAMES[index],
            CATEGORIES[index],
            dueDate,
            100 + i * 15,
            i % 2 == 0
        ));
    }
    System.out.println("Bill Data Generated");
    return bills;
}

public static List<Payment> generatePayments(List<Bill> bills) {
    List<Payment> payments = new ArrayList<>();
    for (int i = 1; i <= 5; i++) {
        Bill bill = bills.get(i - 1);
        LocalDate paymentDate = LocalDate.now().minusDays(i);
        payments.add(new Payment(
            i,
            bill.getBillId(),
            bill.getAmount(),
            paymentDate,
            PAYMENT_MODES[i % PAYMENT_MODES.length]
```

```
        ));
    }
    System.out.println("Payment Data Generated");
    return payments;
  }
}
```

Output Screenshot:

```
PS C:\Users\Sujay R> javac MockDataGenerator.java
PS C:\Users\Sujay R> java MockDataGenerator
User Data Generated
Bill Data Generated
Payment Data Generated
PS C:\Users\Sujay R>
```

TASK 5:
Implement a simple in-memory authentication: Accepts `userId`, `password` and returns true/false

TASK 6: **Yajat Gupta**

If a bill is marked `isRecurring=true`, generate next month's due date and amount on **"bill cycle end"**

Code:

```
// Function to check if a bill is recurring and autogenerating next months's bill
  public static List<Bill> checkAndGenerateRecurringBills(List<Bill> bills)
  {
      List<Bill> newBills = new ArrayList<>();
      Set<Integer> existingIds = new HashSet<>();
```

```
LocalDate currentDate = LocalDate.now();

//Adding existing IDs to the set
for (Bill bill : bills)
{
   existingIds.add(bill.getBillId());
}

for (Bill bill : bills)
{
   if (bill.isRecurring() && bill.getDueDate().isBefore(currentDate))
   {

      // Generating a unique ID that doesn't clash
      int newId;
            do {
                     Random random = new Random();

                newId = random.nextInt(Integer.MAX_VALUE); // Ensures a non-negative
int
            } while (existingIds.contains(newId));

      existingIds.add(newId);

      // Generating Date for the next month's bill
      LocalDate nextDueDate = bill.getDueDate().plusMonths(1);

       //Checking if there is a bill already generated for next month
      if (!hasRecurringBillForNextMonth(bill, bills))
      {
            // Creating bill for the next month
         Bill nextMonthBill = new Bill(
            bill.getUserId(),
            newId,
            bill.getBillName(),
            bill.getCategory(),
            nextDueDate,
            bill.getAmount(),
            true
         );

         newBills.add(nextMonthBill);
      }
   }
```

```
    }

    return newBills;
}
```