GROUP 3

Task Completion Report

Task: Payments Table Creation, Validation, Queries, Triggers.

Task Status

- Current Status: Done
- Work Completed:
    - Created the payments table with constraints and foreign keys.
    - Inserted valid and invalid test records to demonstrate constraints.
    - Created JOIN queries to combine data from users, bills, and payments.
    - Developed views and triggers for business logic enforcement.
    - Implemented Hackathon 1 task with sample code.
    - Output screenshots were captured during testing (attach separately).
    - Final report prepared for GitHub submission to Jahnavi's account.

Part 1: Table Definition

Create payments Table

```
CREATE TABLE payments (
    payment_id          VARCHAR2(50) PRIMARY KEY,
```

```
    payment_date         DATE NOT NULL,
    payment_mode         VARCHAR2(100),
    payer_account_number VARCHAR2(100),
    amount_paid          NUMBER(10, 2),
    status               VARCHAR2(100),
    bill_id              VARCHAR2(50) NOT NULL,
    user_id              VARCHAR2(50) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (bill_id) REFERENCES bills(bill_id),
    CONSTRAINT chk_amount_paid CHECK (amount_paid >= 0)
);
```

OUTPUT:

```
SQL> CREATE TABLE payments (
  2       payment_id VARCHAR2(50) PRIMARY KEY,
  3       payment_date DATE NOT NULL,
  4       payment_mode VARCHAR2(100),
  5       payer_account_number VARCHAR2(100),
  6       amount_paid NUMBER(10, 2),
  7       status VARCHAR2(100),
  8       bill_id VARCHAR2(50) NOT NULL,
  9       user_id VARCHAR2(50) NOT NULL,
 10       FOREIGN KEY (user_id) REFERENCES users(user_id),
 11       FOREIGN KEY (bill_id) REFERENCES bills(bill_id),
 12       CONSTRAINT chk_amount_paid CHECK (amount_paid>=0)
 13   );

Table created.
```

Part 2: Sample Data Insertion

Valid Insert

```
-- Valid: Matches B003 (amount = 2500), user_id = 'U002'
INSERT INTO payments (
    payment_id,
    payment_date,
    payment_mode,
```

```
    payer_account_number,
    amount_paid,
    status,
    bill_id,
    user_id
) VALUES (
    'P001',
    TO_DATE('2025-08-01', 'YYYY-MM-DD'),
    'Credit Card',
    '9876543210',
    2500.00,
    'Completed',
    'B003',
    'U002'
);
```

OUTPUT:

```
SQL> INSERT INTO payments (
  2      payment_id,
  3      payment_date,
  4      payment_mode,
  5      user_id,
  6      payer_account_number,
  7      amount_paid,
  8      status,
  9      bill_id
 10  ) VALUES (
 11      'P001',
 12      TO_DATE('2025-08-01', 'YYYY-MM-DD'),
 13      'Credit Card',
 14      'U002',
 15      '9876543210',
 16      2500.00,
 17      'Completed',
 18      'B003'
 19  );

1 row created.
```

```sql
-- Invalid: amount_paid = 13000 > bill amount = 12000 for B001
INSERT INTO payments (
    payment_id,
    payment_date,
    payment_mode,
    payer_account_number,
    amount_paid,
    status,
    bill_id,
    user_id
) VALUES (
    'P005',
    TO_DATE('2025-08-01', 'YYYY-MM-DD'),
    'Debit Card',
    '9999888877',
    13000.00,
    'Failed',
    'B001',
    'U001'
);
```

```
SQL> INSERT INTO payments (
  2       payment_id,
  3       payment_date,
  4       payment_mode,
  5       user_id,
  6       payer_account_number,
  7       amount_paid,
  8       status,
  9       bill_id
 10   ) VALUES (
 11       'P005',
 12       TO_DATE('2025-08-01', 'YYYY-MM-DD'),
 13       'Debit Card',
 14       'U001',
 15       '9999888877',
 16       13000.00,
 17       'Failed',
 18       'B001'
 19   );

1 row created.
```

Invalid Inserts

(Bill_ID doesn't exist)

```
-- Invalid: bill_id = 'B999' does not exist
INSERT INTO payments (
    payment_id,
    payment_date,
    payment_mode,
    payer_account_number,
    amount_paid,
    status,
    bill_id,
    user_id
) VALUES (
    'P002',
    TO_DATE('2025-08-01', 'YYYY-MM-DD'),
    'Net Banking',
    '1234567890',
    1200.00,
    'Pending',
    'B999',
    'U001'
);
```

OUTPUT:

```
SQL> INSERT INTO payments (
  2      payment_id,
  3      payment_date,
  4      payment_mode,
  5      user_id,
  6      payer_account_number,
  7      amount_paid,
  8      status,
  9      bill_id
 10  ) VALUES (
 11      'P002',
 12      TO_DATE('2025-08-01', 'YYYY-MM-DD'),
 13      'Net Banking',
 14      'U001',
 15      '1234567890',
 16      1200.00,
 17      'Pending',
 18      'B999'
 19  );
INSERT INTO payments (
*
ERROR at line 1:
ORA-02291: integrity constraint (PALAK.SYS_C008376) violated - parent key not
found
```

(User_ID doesn't exist)

```sql
-- Invalid: user_id = 'U999' does not exist
INSERT INTO payments (
    payment_id,
    payment_date,
    payment_mode,
    payer_account_number,
    amount_paid,
    status,
    bill_id,
    user_id
) VALUES (
    'P003',
    TO_DATE('2025-08-01', 'YYYY-MM-DD'),
    'UPI',
    '1122334455',
    1200.00,
    'Failed',
    'B001',
    'U999'
);
```

```
SQL> INSERT INTO payments (
  2      payment_id,
  3      payment_date,
  4      payment_mode,
  5      user_id,
  6      payer_account_number,
  7      amount_paid,
  8      status,
  9      bill_id
 10  ) VALUES (
 11      'P003',
 12      TO_DATE('2025-08-01', 'YYYY-MM-DD'),
 13      'UPI',
 14      'U999',
 15      '1122334455',
 16      1200.00,
 17      'Failed',
 18      'B001'
 19  );
INSERT INTO payments (
*
ERROR at line 1:
ORA-02291: integrity constraint (PALAK.SYS_C008375) violated - parent key not
found
```

(Violates CHECK constraint)

```
-- Invalid: amount_paid = -500 (violates CHECK constraint)
INSERT INTO payments (
    payment_id,
    payment_date,
    payment_mode,
    payer_account_number,
    amount_paid,
    status,
    bill_id,
    user_id
) VALUES (
    'P004',
    TO_DATE('2025-08-01', 'YYYY-MM-DD'),
    'Cash',
    '0000111122',
    -500.00,
    'Failed',
    'B001',
    'U001'
```

```
);
```

```
SQL>
SQL> INSERT INTO payments (
  2      payment_id,
  3      payment_date,
  4      payment_mode,
  5      user_id,
  6      payer_account_number,
  7      amount_paid,
  8      status,
  9      bill_id
 10  ) VALUES (
 11      'P004',
 12      TO_DATE('2025-08-01', 'YYYY-MM-DD'),
 13      'Cash',
 14      'U001',
 15      '0000111122',
 16      -500.00,
 17      'Failed',
 18      'B001'
 19  );
INSERT INTO payments (
*
ERROR at line 1:
ORA-02290: check constraint (PALAK.CHK_AMOUNT_PAID) violated
```

Part 3: SQL Queries and Views

Join Query: Users, Bills, Payments

```sql
SELECT
    u.name AS user_name,
    b.bill_category,
    b.amount,
    p.payment_date
FROM
    users u
JOIN
    bills b ON u.user_id = b.user_id
```

```
JOIN
    payments p ON b.bill_id = p.bill_id;
```

OUTPUT:

```
SQL> SELECT
  2      u.name AS user_name,
  3      b.bill_category,
  4      b.amount,
  5      p.payment_date
  6  FROM
  7      users u
  8  JOIN
  9      bills b ON u.user_id = b.user_id
 10  JOIN
 11      payments p ON b.bill_id = p.bill_id;

USER_NAME
--------------------------------------------------------------------------------
BILL_CATEGORY                                       AMOUNT PAYMENT_D
--------------------------------------------- ---------- ---------
Priya Sharma
electricity                                         2500 01-AUG-25

Sanjay Kumar
rent                                               12000 01-AUG-25
```

Description: Displays user name, bill category, bill amount, and payment date.

Part 4: Create View for fully paid bills

```
CREATE VIEW fully_paid_bills AS
SELECT
  u.name,
  b.bill_id,
  b.bill_category,
  b.amount,
  p.amount_paid,
  p.payment_date
FROM bills b
JOIN users u ON b.user_id = u.user_id
JOIN payments p ON b.bill_id = p.bill_id
WHERE b.is_paid = 1;
```

Description: View lists all bills that have been fully paid.

```
SQL> CREATE VIEW fully_paid_bills AS
  2  SELECT
  3    u.name,
  4    b.bill_id,
  5    b.bill_category,
  6    b.amount,
  7    p.amount_paid,
  8    p.payment_date
  9  FROM bills b
 10  JOIN users u ON b.user_id = u.user_id
 11  JOIN payments p ON b.bill_id = p.bill_id
 12  WHERE b.is_paid = 1;

View created.
```

Part 5: Triggers

Trigger: Update is_paid After Sufficient Payment

```
CREATE OR REPLACE TRIGGER trg_update_bill_status_after_payment
AFTER INSERT ON payments
FOR EACH ROW
DECLARE
    v_total_paid  NUMBER := 0;
    v_bill_amount NUMBER := 0;
BEGIN
    SELECT NVL(SUM(amount_paid), 0)
    INTO v_total_paid
    FROM payments
    WHERE bill_id = :NEW.bill_id;

    SELECT amount
    INTO v_bill_amount
    FROM bills
```

```
    WHERE bill_id = :NEW.bill_id;

    IF v_total_paid >= v_bill_amount THEN
        UPDATE bills
        SET is_paid = 1
        WHERE bill_id = :NEW.bill_id;
    END IF;
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER trg_update_bill_status_after_payment
  2  AFTER INSERT ON payments
  3  FOR EACH ROW
  4  DECLARE
  5      v_total_paid  NUMBER := 0;
  6      v_bill_amount NUMBER := 0;
  7  BEGIN
  8
  9      SELECT NVL(SUM(amount_paid), 0)
 10      INTO v_total_paid
 11      FROM payments
 12      WHERE bill_id = :NEW.bill_id;
 13
 14
 15      SELECT amount
 16      INTO v_bill_amount
 17      FROM bills
 18      WHERE bill_id = :NEW.bill_id;
 19
 20
 21      IF v_total_paid >= v_bill_amount THEN
 22          UPDATE bills
 23          SET is_paid = 1
 24          WHERE bill_id = :NEW.bill_id;
 25      END IF;
 26  END;
 27  /

Trigger created.
```

Description: Sets is_paid = 1 on the bill when total payment meets or exceeds bill amount.

Part 6: Add a check to ensure amount paid <= Total amount

```
CREATE OR REPLACE TRIGGER trg_check_payment_amount
BEFORE INSERT OR UPDATE ON payments
FOR EACH ROW
DECLARE
    v_bill_amount NUMBER(10,2);
BEGIN
    SELECT amount INTO v_bill_amount FROM bills WHERE bill_id =
:NEW.bill_id;

    IF :NEW.amount_paid > v_bill_amount THEN
        RAISE_APPLICATION_ERROR(-20001, 'Payment amount cannot be greater
than bill amount.');
    END IF;
END;
```

```
SQL> CREATE OR REPLACE TRIGGER trg_check_payment_amount
  2  BEFORE INSERT OR UPDATE ON payments
  3  FOR EACH ROW
  4  DECLARE
  5      v_bill_amount NUMBER(10,2);
  6  BEGIN
  7      SELECT amount INTO v_bill_amount FROM bills WHERE bill_id = :NEW.bill_id;
  8
  9      IF :NEW.amount_paid > v_bill_amount THEN
 10          RAISE_APPLICATION_ERROR(-20001, 'Payment amount cannot be greater than bill amount
.');
 11      END IF;
 12  END;
 13  /

Trigger created.
```

Description: Throws error if payment amount exceeds the original bill amount.

Part 7: PNB-29

Problem: **Write a query to show total due amount per user. Group by user.**

```
SELECT
    u.name AS "User Name",
```

```
    SUM(b.amount - NVL(p.total_paid_per_bill, 0)) AS "Total Due Amount"
FROM
    users u
INNER JOIN
    bills b ON u.user_id = b.user_id
LEFT JOIN
    (
        SELECT
            bill_id,
            SUM(amount_paid) AS total_paid_per_bill
        FROM
            payments
        GROUP BY
            bill_id
    ) p ON b.bill_id = p.bill_id
GROUP BY
    u.name
ORDER BY
    "Total Due Amount" DESC;
```

Part 8: Hackathon Task Summary

Description

Design and implement a small payment management system including:

- Tables with appropriate constraints
- Business logic enforcement via triggers
- Reporting via joins and views

- Validation for payment integrity

Deliverables

- Table: payments
- Triggers:
    - trg_check_payment_amount
    - trg_update_bill_status_after_payment
- View: fully_paid_bills
- SQL Queries: For billing summary and validation
- Screenshots: To be attached manually (outputs, errors, views)

Submission Instructions

- The task has been completed and verified.
- Final report and SQL code will be pushed to Jahnavi's GitHub repository.
- Screenshots will be attached in the GitHub folder.
- Requesting final review and closure of the task.