CODE

```
mysql> drop database if exists A5;
mysql> DROP PROCEDURE IF EXISTS setFine;

mysql> create database A5;
Query OK, 1 row affected (0.00 sec)

mysql> use A5;
Database changed
mysql> DROP PROCEDURE IF EXISTS setFine;

Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql>
mysql> create table Customer(
   ->                Cust_id int not null,
   ->                Name varchar(30),
   ->                DateOfPayment date,
   ->                NameOfScheme varchar(20),
   ->                Status varchar(10),
   ->                primary key(Cust_id)
   ->                );

mysql>
mysql> create table Fine(
   ->                Cust_id int not null,
   ->                Date date,
   ->                Amt int,
   ->                foreign key(Cust_id) references Customer(Cust_id) on delete cascade
   ->                );
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> insert into Customer VALUES(1, "Prathamesh", "2020-04-8", "High-return", "N" );

mysql> insert into Customer VALUES(2, "Aditya", "2020-03-15", "Low-return", "N" );
Query OK, 1 row affected (0.00 sec)

mysql> insert into Customer VALUES(3, "Sourav", "2020-03-12", "High-return", "N" );
Query OK, 1 row affected (0.00 sec)

mysql> insert into Customer VALUES(4, "Rajesh", "2020-03-1", "Low-return", "N" );
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into Customer VALUES(5, "Suman", "2020-03-27", "Low-return", "N" );
Query OK, 1 row affected (0.01 sec)

mysql>
mysql>
mysql> delimiter @@
mysql> select * from Customer@@
+---------+------------+--------------+--------------+--------+
| Cust_id | Name       | DateOfPayment | NameOfScheme | Status |
+---------+------------+--------------+--------------+--------+
|       1 | Prathamesh | 2020-04-08    | High-return  | N      |
|       2 | Aditya     | 2020-03-15    | Low-return   | N      |
|       3 | Sourav     | 2020-03-12    | High-return  | N      |
|       4 | Rajesh     | 2020-03-01    | Low-return   | N      |
|       5 | Suman      | 2020-03-27    | Low-return   | N      |
+---------+------------+--------------+--------------+--------+
5 rows in set (0.00 sec)

mysql> create PROCEDURE setFine(IN id int, IN NameOfScheme varchar(20))
    -> BEGIN
    ->     declare myFine INT;
    ->     declare myDate date;
    ->     declare myStatus VARCHAR(10);
    ->     declare days int;
    ->     declare diff int;
    ->     declare exit handler for 1062
    ->     select 'Error : Duplicate' as message;
    ->     declare exit handler for not found
    ->     select 'Error :  Record not found' as message;
    ->     select DateOfPayment into myDate FROM Customer where Cust_id = id;
    ->     SELECT Status into myStatus FROM Customer where Cust_id = id;
    ->     select DATEDIFF(CURDATE() , myDate) into diff;
    ->
    ->     IF myStatus="N" THEN
    ->         IF diff>15 AND diff<=30 THEN
    ->             set myFine = 5*diff;
    ->         END IF;
    ->         IF diff>30 THEN
    ->             set myFine = 50*(diff-30) + 75;
    ->         END IF;
    ->
    ->         INSERT INTO Fine VALUES(id, myDate, myFine);
    ->         UPDATE Customer set Status="P" where Cust_id = id;
    ->
    ->     END IF;
    -> END @@
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter ;
```

```
mysql>
mysql>

mysql> call setFine(1, "High-return");
Query OK, 1 row affected (0.03 sec)

mysql> select * from Fine;
+---------+------------+------+
| Cust_id | Date       | Amt  |
+---------+------------+------+
|       1 | 2020-04-08 | 6525 |
+---------+------------+------+
1 row in set (0.00 sec)

mysql> select * from Customer;
+---------+------------+---------------+--------------+--------+
| Cust_id | Name       | DateOfPayment | NameOfScheme | Status |
+---------+------------+---------------+--------------+--------+
|       1 | Prathamesh | 2020-04-08    | High-return  | P      |
|       2 | Aditya     | 2020-03-15    | Low-return   | N      |
|       3 | Sourav     | 2020-03-12    | High-return  | N      |
|       4 | Rajesh     | 2020-03-01    | Low-return   | N      |
|       5 | Suman      | 2020-03-27    | Low-return   | N      |
+---------+------------+---------------+--------------+--------+
5 rows in set (0.00 sec)

mysql> call setFine(11, "High-return");
+--------------------------+
| message                  |
+--------------------------+
| Error :  Record not found |
+--------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

Assignment 5

- Title: PL/SQL block
- Date of Completion: 9/9/20
- Date of Submission: 30/9/20
- Problem Statement:
Write a PL/SQL block of code for the following requirements.
Schema:
Customer (Cust_id, Name, Date of Payment, Name_of_scheme, status)
Fine (Cust-id, Date, Amt)

1. Accept Cust-id & name of scheme
2. Check no. of days from date of payment
   a) It between 15-30 days : Fine 5/day
   b) Greater than 30 days : Fine 50/day
3. After payment status changed from N to P
4. If condition of fine is true, add into Fine table.

- Objective:
1) To understand PL/SQL block
2) To understand exception handling.
3) To apply control structure.

- Outcome:
  Students will be able to apply PL/SQL block, user defined & predefined exception handling.

- S/W & H/W Requirements:
  Windows 10, MySQL, i5 processor, keyboard, mouse.

- Theory:

★ ORACLE

- PL/SQL -
  It stands for procedural language/ Structured Query language, PL/SQL offers a set of procedural commands (IF statements, loops, assignments), organised within blocks that complement & extend the reach of SQL.

- Blocks of PL/SQL:-
  A block is defined by the keywords DECLARE, BEGIN, EXCEPTION & END which breakup the block into 3 sections.

1) Declarative Statements:
That declare constants, variables, & other code elements.

2) Executable Statements:
That are run when block is executed

3) Exception handling:
A section you can use to catch or trap any exceptions.

- Exception Handling:
Exception message consists of 3 parts:
1) Type of Exception
2) An error code
3) message.

Syntax:

```
DECLARE
    declare section
BEGIN
    EXCEPTION
        WHEN ex1 THEN
            - error handling
        WHEN ex2 THEN
            - error handling
END
```

- Types of Exceptions:
  1) Named System Exceptions
     Eg: No data found, Zero-divide
  2) UnNamed System Exceptions
     a) Oracle provides no name for these
     b) Come with a code & message
     c) We can assign names to this using Exception INIT.
  3) User defined Exceptions
     a) We can define these based on our business rules.
     b) They should be declared & raised explicitly in PL/SQL block

- Control Structure:
  The selection structure selects a block of statements to run based on a certain condition

  Syntax:

```
DECLARE
   -
BEGIN
   IF  cond"  THEN
          statements
   ELSE
          Statements
   END IF ;
END ;
```

- MYSQL
  Stored Procedures (Subroutine / Sub Program)

  Syntax:

```
Create procedure procedure_name (parameters)
BEGIN
DECLARE
    - declaration
    - Execution
END;
```

- Dropping a procedure
  DROP procedure procedure_name

- Parameters:
  IN - passing parameters when calling
  OUT - value of parameter changes in procedure & a processed output is received.
  INOUT - pass initial value change it & then receive it.

- Calling a Procedure:
  Syntax:
  call procedure_name (parameters)

- TEST CASES:

	input	expected output	Status
1)	call proc1 (1, 'recurring')	1, 2020-09-09 Amt = 150	Yes
2)	call proc1 (4, 'fixed')	customer has already paid	Yes

- Conclusion:
Thus we have learnt about PL/SQL block, exception handling & implemented the same in MYSQL