Wayne Construction Company

DATABASE MODEL

Shruti Shetty 001850949

Contents

1. Description and Purpose

The database is for **WAYNE CONSTRUCTION's** company. This database will help the company to manage their employees and the customers that have booked a flat with the company. It will help them to keep a track of all payments and invoices made by the customers for the flat. The company will also be able to keep a track of all payments to be made to the suppliers and vendors for all equipment's. A database management system will be extremely useful for the company to keep a track of the above activities in an efficient manner. Following are the cases that would be considered for the company

1. A list of all customers that have booked the flat with the company
2. Details of employees working for the organization
3. Company's buildings also to be maintained in the database
4. A list of all suppliers, vendors and their respective equipment's
5. An order list that the company places to pay its providers and suppliers
6. Keeping a track of payments done by the customer
7. Maintaining the bank account details for the company
8. Modes of payment (Cash, Cheque or an online transfer, Auto)
9. Invoice Authorization Options (Parallel, Serial or Normal)
10. Roles of the employees (Verifier and Authorization)
11. Instalment payment details that are to be made by the customer

2. Table Description

1. Customer
   This table will contain all details of the customer that have booked a flat with the construction company

2. Accounts
   The table will have a list of accounts that the customer will make payment from.

3. Flats
   Flat Table will consist a list of flats that the customer will book. It will contain details of the flat like the Room No, Floor No, Floor Plan etc.

4. Orders
   This table will consist the Order Details that the Customer places for booking a flat.

5. Building
   It will contain a list of Buildings that the Wayne Construction Company has constructed or if it's under construction.

6. Builds
   It's a bridge table between Building and Employees which will give information as to which employee is responsible for the construction of the building.

7. Employees
   This will have a list of employees which work for the Wayne Construction Company.

8. Invoices
   Every payment made by the customer will be done against an Invoice. This table will store information about these invoices.

9. Financial_Transaction
   This table will contain details of the Financial Transactions that will be made by the customer. This will help the company to calculate how much pending amount is left for the customer to make for the flat.

10. Payments
    This will contain payment information that the customer makes.

11. Transaction_Type
    Transaction made by the customer against each payment will have a type whether the customer wants to make Online payment, Cash Payment or does he want to have Auto Payment.

12. Supplies
    The company needs to have details of the list of Equipment's it has ordered from all Manufacturers. It's basically a bridge table between Manufacturer and Employee of the company.

13. Manufacturer
    List of Manufacturers that the construction company deals with in order to get their Supplies.

14. Equipments_has_Manufacturer
    It's a bridge table between the Manufacturer and Equipment which will have details as to which Manufacturer owns which Equipment.

15. Equipment
    A list of all equipment's.

3. List of Relationships

1. Customer has many Accounts: A customer can make payments for the flat from the multiple accounts he/she has
2. Customer has many Orders: A customer can place many Orders to book multiple flats. Each Flat booking should correspond to an Order
3. Building has many Flats: Any building constructed by the company will consist of many flats
4. Customer has many Payments: The customer can pay the flat amount in instalments as and when the building's construction is progressing
5. Manufacture has many Equipment: Manufacturer has an inventory of Equipment's

4. ER Diagram

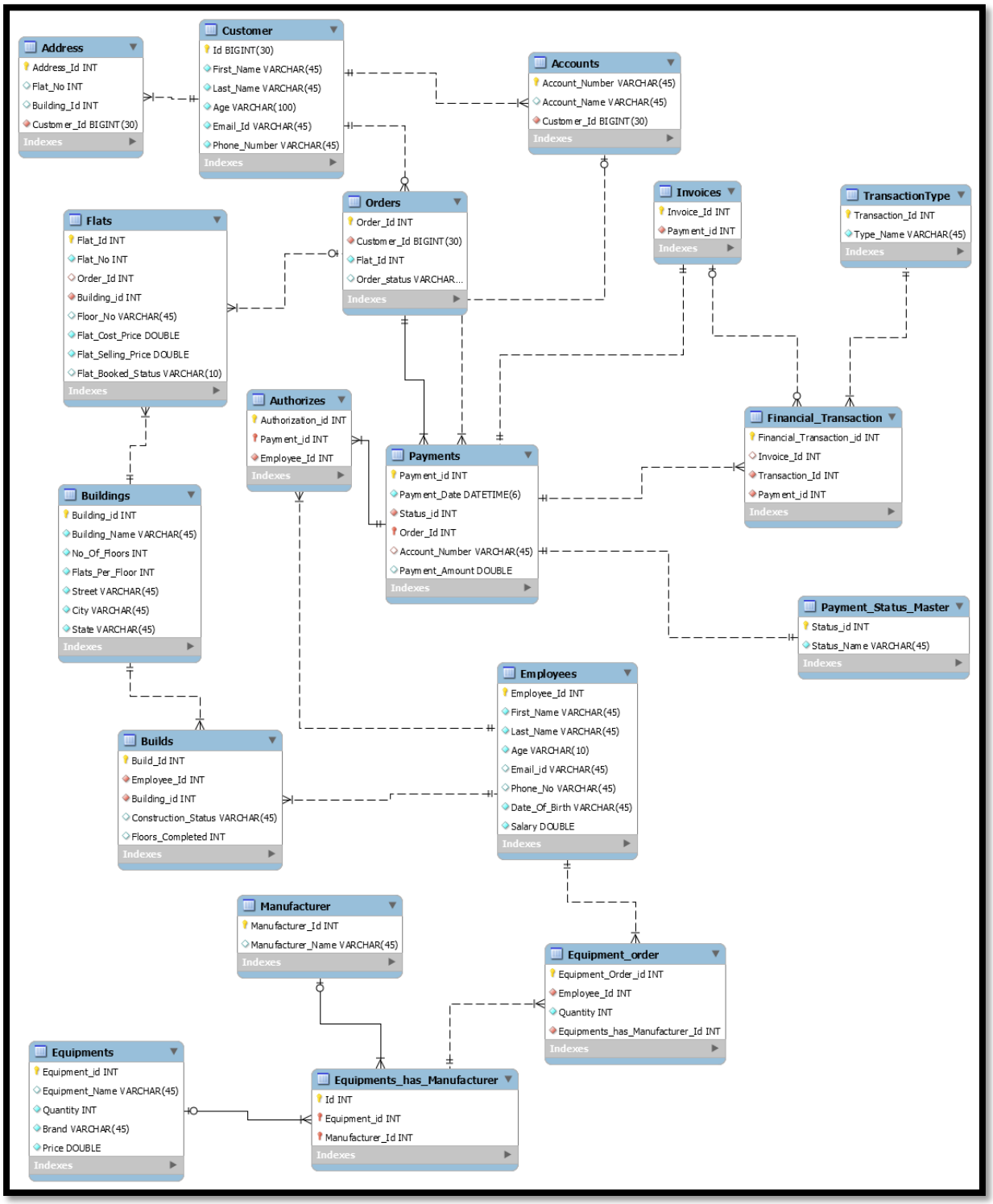


Fig 4.1 EER Diagram for Wayne Construction Company

5. Create Table Script

This script consists of creation of database **mydb** and all tables that are present in the database.

CreateScript.sql

6. Use Cases

I. Customer books flats **SP_PLACE_ORDERS**
II. Customer Cancels an Order **SP_CANCEL_ORDER**
III. Once the Customer books a flat he initiates a payment for the flat **SP_MAKE_PAYMENT_BY_CUSTOMER**
IV. Authorization of Payment By Employee **SP_AUTHORIZE_PAYMENTS**
V. Employee builds a building **SP_INSERT_INTO_BUILDS**
VI. Flat Details that the Customer has booked. **VW_CUSTOMER_FLAT_DETAILS**
VII. No. of Payments made by the Customer for the Flat booked. **VW_CUSTOMER_FLAT_PAYMENT_DETAILS**
VIII. Balance Amount to be paid by the Customer for the Flat. **FN_GET_BALANCE_AMOUNT (Customer Id, flat Id)**
IX. Construction Status of the Building (ie. Remaining No of Floors to be constructed) **Error! Reference source not found.**
X. Address of the Customer (ie. If Payment is completed update address of the table) **Error! Reference source not found.**
XI. Total Equipment's ordered by the Company **VW_GET_EQUIPMENT_ORDER_DETAILS**
XII. When a building record is inserted in a building table. Corresponding flat entries will be inserted by using **TR_AFTER_BUILDING_INSERT** trigger.
XIII. Total Revenue of the Company **SP_GET_TOTAL_PROFIT**

7. Users And Privileges.

There are three following type of users.

1. root
   This will act as the administrator with full access to database.

2. Employee
   Employee user will have limited access to database. An employee will be able to view certain tables with limited columns.

Employee
Tables
buildings
Customers
Employees (employeed id, first Name,Last Name, Date of Birth)
Equipments
Equipment Has Manufacturer

Flats
orders
supplies
Views
VW_GET_STATUS_OF_BUILDING
VW_GET_CUSTOMER_ADDRESS
VW_GET_EQUIPMENT_ORDER_DETAILS
VW_CUSTOMER_FLAT_DETAILS
Procedures
SP_AUTHORIZE_PAYMENTS
SP_GET_TOTAL_PROFIT
SP_INSERT_INTO_BUILDS
SP_PLACE_EQUIPMENT_ORDER

Grant select Privileges to Employee

```
mysql> grant select on buildings to 'emp1'@'localhost';
Query OK, 0 rows affected (0.11 sec)

mysql> grant select on Customer to 'emp1'@'localhost';
Query OK, 0 rows affected (0.06 sec)

mysql> grant select(employee_id, first_Name, last_Name, Date_Of_birth) on Employees to 'emp1'@'localhost';
Query OK, 0 rows affected (0.14 sec)

mysql> create select on Equipments to 'emp1'@'localhost';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server ve
 for the right syntax to use near 'select on Equipments to 'emp1'@'localhost'' at line 1
mysql> grant select on Equipments to 'emp1'@'localhost';
Query OK, 0 rows affected (0.05 sec)

mysql> grant se^C
mysql> grant select on Equipment_Has_Manufacturer to 'emp1'@'localhost';
ERROR 1146 (42S02): Table 'mydb.Equipment_Has_Manufacturer' doesn't exist
mysql> grant select on Equipments_Has_Manufacturer to 'emp1'@'localhost';
Query OK, 0 rows affected (0.04 sec)

mysql> grant select on Flats to 'emp1'@'localhost';
Query OK, 0 rows affected (0.04 sec)

mysql> grant select on Orders to 'emp1'@'localhost';
Query OK, 0 rows affected (0.05 sec)
```

Grant Select on View Privileges to Employee

```
mysql> grant select on mydb.VW_GET_STATUS_OF_BUILDING to 'emp1'@'localhost';
Query OK, 0 rows affected (0.04 sec)

mysql> grant select on mydb.VW_GET_CUSTOMER_ADDRESS to 'emp1'@'localhost';
ERROR 1146 (42S02): Table 'mydb.VW_GET_CUSTOMER_ADDRESS' doesn't exist
mysql> grant select on mydb.VW_GET_CUSTOMER_ADDRESS to 'emp1'@'localhost';
Query OK, 0 rows affected (0.04 sec)

mysql> grant select on mydb.VW_GET_EQUIPMENT_ORDER_DETAILS to 'emp1'@'localhost';
Query OK, 0 rows affected (0.05 sec)

mysql> grant select on mydb.VW_CUSTOMER_FLAT_DETAILS to 'emp1'@'localhost';
Query OK, 0 rows affected (0.08 sec)
```

Grant execute privileges to emp1

```
mysql> grant execute on procedure mydb.SP_AUTHORIZE_PAYMENTS to 'emp1'@'localhost';
Query OK, 0 rows affected (0.11 sec)

mysql> grant execute on procedure mydb.SP_GET_TOTAL_PROFIT to 'emp1'@'localhost';
Query OK, 0 rows affected (0.10 sec)

mysql> grant execute on procedure mydb.SP_INSERT_INTO_BUILDS to 'emp1'@'localhost';
Query OK, 0 rows affected (0.08 sec)

mysql> grant execute on procedure mydb.SP_PLACE_EQUIPMENT_ORDER to 'emp1'@'localhost';
Query OK, 0 rows affected (0.04 sec)
```

3. Customer

   Customer will have the least access to database as compared to Employee and Customer

Customer
Tables
buildings
Customer (Id, FirstName, LastName)
Flats
Views
VW_GET_STATUS_OF_BUILDING
VW_GET_CUSTOMER_ADDRESS_FOR_CUSTOMER
VW_CUSTOMER_FLAT_DETAILS with limited access
VW_CUSTOMER_FLAT_PAYMENT_DETAILS
Procedres
SP_CANCEL_ORDER
SP_MAKE_PAYMENT_BY_CUSTOMERS
SP_PLACE_ORDERS

Grant Procedure privileges to Customers

```
mysql> use mydb;
Database changed
mysql> create user 'Shruti'@'localhost' identified by 'Shruti';
Query OK, 0 rows affected (0.12 sec)

mysql> grant execute on procedure mydb.SP_CANCEL_ORDER to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.11 sec)

mysql> grant execute on procedure mydb.SP_MAKE_PAYMENT to 'Shruti'@'localhost';
ERROR 1305 (42000): PROCEDURE sp_make_payment does not exist
mysql> grant execute on procedure mydb.SP_MAKE_PAYMENT_BY_CUSTOMERS to 'Shruti'@'localhost';
ERROR 1305 (42000): PROCEDURE sp_make_payment_by_customers does not exist
mysql> grant execute on procedure mydb.SP_MAKE_PAYMENT_BY_CUSTOMER to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.19 sec)

mysql> grant execute on procedure mydb.SP_PLACE_ORDERS to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.05 sec)
```

Grant select privileges on Views

```
mysql> grant select on mydb.VW_GET_STATUS_OF_BUILDING to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.07 sec)

mysql> grant select on mydb.VW_CUSTOMER_FLAT_PAYMENT_DETAILS to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.05 sec)

mysql> grant select on mydb.VW_GET_CUSTOMER_ADDRESS_FOR_CUSTOMER to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.05 sec)

mysql> grant select on mydb.VW_CUSTOMER_FLAT_DETAILS_FOR_CUSTOMER to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.05 sec)
```

Grant select privileges on Tables

```
mysql> grant select on buildings to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.05 sec)

mysql> grant select on flats to 'Shruti'@'localhost';
Query OK, 0 rows affected (0.05 sec)
```

8. Functions

1. FN_GET_BALANCE_AMOUNT (Customer Id, flat Id)

This function gives the balance amount that a Customer will have to pay for the flat that he has booked.

```
DROP FUNCTION IF EXISTS MYDB.FN_GET_BALANCE_AMOUNT;

DELIMITER $$
CREATE FUNCTION MYDB.FN_GET_BALANCE_AMOUNT (V_CUST_ID BIGINT(30),  V_FLAT_ID INT) RETURNS DOUBLE
    DETERMINISTIC
BEGIN
    DECLARE V_BALANCE_AMOUNT DOUBLE;
    DECLARE V_TOTAL_AMOUNT DOUBLE;
    DECLARE V_SELLING_PRICE DOUBLE;

    SELECT SUM(PAYMENT_AMOUNT) INTO V_TOTAL_AMOUNT FROM MYDB.VW_CUSTOMER_FLAT_PAYMENT_DETAILS
    WHERE FLAT_BOOKED = V_FLAT_ID AND ID = V_CUST_ID;

    SELECT FLAT_SELLING_PRICE INTO V_SELLING_PRICE FROM MYDB.FLATS
    WHERE FLATS.FLAT_ID = V_FLAT_ID;

    SET V_BALANCE_AMOUNT = V_SELLING_PRICE - V_TOTAL_AMOUNT;

    RETURN (V_BALANCE_AMOUNT);
END $$
DELIMITER ;
```

```
-- AUTHORIZE PAYMENT BY EMPLOYEE

CALL SP_AUTHORIZE_PAYMENTS(24,2,20); -- PAYMENT_ID, EMPLOYEE_ID, PAYMENT_STATUS

SELECT * FROM INVOICES;

SELECT * FROM PAYMENTS; -- PAYMENTS STATUS UPDATED

SELECT * FROM FINANCIAL_TRANSACTION;

-- GET BALANCE AMOUNT
SELECT FN_GET_BALANCE_AMOUNT(6,103); -- CUSTOMER_ID, FLAT_ID
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

FN_GET_BALANCE_AMOUNT(6,103)
1050

2. FN_GET_CONSTRUCTION_STATUS

This function will give the construction status of the building. This function is used inside a Stored Procedure **SP_INSERT_INTO_BUILDS** to insert construction status of the building. It takes input as Floors completed and Total Floors in the building.

Limit to 1000 rows

```
DROP FUNCTION IF EXISTS MYDB.FN_GET_CONSTRUCTION_STATUS;

DELIMITER $$
    CREATE FUNCTION MYDB.FN_GET_CONSTRUCTION_STATUS(V_FLOOR INT, V_TOTAL_FLOORS INT) RETURNS VARCHAR(45)
        DETERMINISTIC
    BEGIN
        DECLARE V_CONS_STATUS VARCHAR(45);

        IF V_FLOOR = 0 THEN
            SET V_CONS_STATUS = 'Initiated';
        ELSEIF V_FLOOR < V_TOTAL_FLOORS THEN
            SET V_CONS_STATUS = 'In Progress';
        ELSEIF V_FLOOR = V_TOTAL_FLOORS THEN
            SET V_CONS_STATUS = 'Completed';
        END IF;
    RETURN V_CONS_STATUS;
    END $$
DELIMITER ;
```

9. Views

1. VW_GET_STATUS_OF_BUILDING

This view will give the construction status of all the buildings that are constructed by the company. This view will provide details such as Building Id, Building Name, Construction Status, (Initiated, In Progress or Completed) Total Floors, No of Floors left for construction

```sql
CREATE VIEW MYDB.VW_GET_STATUS_OF_BUILDING AS
SELECT BUILDINGS.BUILDING_ID, BUILDINGS.BUILDING_NAME,
BUILDS.CONSTRUCTION_STATUS AS 'STATUS', BUILDS.FLOORS_COMPLETED, BUILDINGS.NO_OF_FLOORS,
(BUILDINGS.NO_OF_FLOORS - BUILDS.FLOORS_COMPLETED) AS 'LEFT TO BUILD'
FROM BUILDS, BUILDINGS
WHERE BUILDINGS.BUILDING_ID = BUILDS.BUILDING_ID
WITH CHECK OPTION;


#DROP VIEW MYDB.VW_GET_STATUS_OF_BUILDING;
```

SQL File 25* | SQL File 31* | Buildings | PROJECT_SCRIPT | TR_AFTER_BUILDING_INSERT | SP_PLACE_ORDERS - Routine

```sql
SELECT * FROM MYDB.VW_GET_STATUS_OF_BUILDING;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

BUILDING_ID	BUILDING_NAME	STATUS	FLOORS_COMPLETED	NO_OF_FLOORS	LEFT TO BUILD
1	LANDMARK SQUARE APT	Initiated	0	9	9
2	YOGI CHHAYA	In Progress	3	10	7
3	CUSTOM HOUSE TOWER	In Progress	3	12	9

2. VW_GET_EQUIPMENT_ORDER_DETAILS

This view will give details about equipment and the orders placed by the employee. It will have the manufacturer name, equipment name, quantity for which order was placed and the total price. So, if someone wants to view order details made by the company for the construction process.

```sql
CREATE VIEW MYDB.VW_GET_EQUIPMENT_ORDER_DETAILS AS
SELECT EQUIPMENT_ORDERS.EQUIPMENT_ORDERS_ID, EQUIPMENT_ORDERS.EMPLOYEE_ID,
MANUFACTURER.MANUFACTURER_NAME, EQUIPMENTS.EQUIPMENT_NAME,
EQUIPMENT_ORDERS.QUANTITY,EQUIPMENTS.PRICE AS 'INDIVIDUAL PRICE',
(EQUIPMENT_ORDERS.QUANTITY * EQUIPMENTS.PRICE) AS 'TOTAL_PRICE'
FROM EQUIPMENT_ORDERS INNER JOIN EQUIPMENTS_HAS_MANUFACTURER
ON EQUIPMENTS_HAS_MANUFACTURER.ID = EQUIPMENT_ORDERS.EQUIPMENT_HAS_MANUFACTURER_ID
INNER JOIN MANUFACTURER
ON MANUFACTURER.MANUFACTURER_ID = EQUIPMENTS_HAS_MANUFACTURER.MANUFACTURER_ID
INNER JOIN EQUIPMENTS
ON EQUIPMENTS.EQUIPMENT_ID = EQUIPMENTS_HAS_MANUFACTURER.EQUIPMENT_ID
WITH CHECK OPTION;

#DROP VIEW MYDB.VW_GET_EQUIPMENT_ORDER_DETAILS
```

SQL File 25* | SQL File 31* | Buildings | PROJECT_SCRIPT | TR_AFTER_BUILDING_INSERT | SP_PLACE_ORDERS - Routine

Limit to 1000 rows

```
SELECT * FROM MYDB.VW_GET_EQUIPMENT_ORDER_DETAILS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EQUIPMENT_ORDERS_ID	EMPLOYEE_ID	MANUFACTURER_NAME	EQUIPMENT_NAME	QUANTITY	INDIVIDUAL PRICE	TOTAL_PRICE
2	1	RAHUL TITARE	EXCAVATORS	10	400	4000
3	2	KAMLESSH JAIWAL	BACKHOE LOADERS	1	300	300
4	2	KAMLESSH JAIWAL	BACKHOE LOADERS	1	300	300
5	2	KAMLESSH JAIWAL	BACKHOE LOADERS	1	300	300

3. VW_CUSTOMER_FLAT_DETAILS

This view is to know the Customer booked flat details.

SQL File 25* | SQL File 31* | Buildings | PROJECT_SCRIPT | TR_AFTER_BUILDING_INSERT | SP_PLACE_ORDERS - Routine | VW_CUSTOI

Limit to 1000 rows

```
## TO VIEW CUSTOMER BOOKED FLAT DETAILS

CREATE VIEW MYDB.VW_CUSTOMER_FLAT_DETAILS AS
SELECT  CUSTOMER.ID, CUSTOMER.FIRST_NAME, CUSTOMER.LAST_NAME, FLATS.FLAT_NO,
BUILDINGS.BUILDING_ID, BUILDINGS.BUILDING_NAME, BUILDINGS.STREET, BUILDINGS.CITY, BUILDINGS.STATE
,ORDERS.FLAT_ID
FROM MYDB.CUSTOMER
INNER JOIN MYDB.ORDERS
ON CUSTOMER.ID = ORDERS.CUSTOMER_ID
INNER JOIN MYDB.FLATS
ON ORDERS.FLAT_ID = FLATS.FLAT_ID
INNER JOIN MYDB.BUILDINGS
ON FLATS.BUILDING_ID = BUILDINGS.BUILDING_ID
WITH CHECK OPTION;

#DROP VIEW MYDB.VW_CUSTOMER_FLAT_DETAILS;
```

SQL File 25* | SQL File 31* | Buildings | PROJECT_SCRIPT | TR_AFTER_BUILDING_INSERT | SP_PLACE_ORDERS - Routine

Limit to 1000 rows

```
SELECT * FROM MYDB.VW_CUSTOMER_FLAT_DETAILS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	FIRST_NAME	LAST_NAME	FLAT_NO	BUILDING_ID	BUILDING_NAME	STREET	CITY	STATE	FLAT_ID
2	Swathi	Shetty	2	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	2
1	Shruti	Shetty	1	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	1
1	Shruti	Shetty	4	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	4
2	Swathi	Shetty	3	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	3
1	Shruti	Shetty	10	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	10
2	Swathi	Shetty	12	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	12
3	Arjun	Shetty	13	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	13
4	Satisha	Shetty	16	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	16
5	Prema	Shetty	17	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	17
6	Siddharth	Kotian	18	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	18
1	Shruti	Shetty	20	1	LANDMARK SQUARE APT	PETERBOROUGH ST.	BOSTON	MA	20

VW_CUSTOMER_FLAT_DETAIL... | Read Only

4. VW_CUSTOMER_FLAT_PAYMENT_DETAILS_FOR_ADMIN

This view consists of all payment details that the customer has done to book his flat. It will contain the instalment details such as Order Id, Payment Id, Payment Date, Account Number. Amount paid etc.

PROJECT_SCRIPT | SQL File 25* | SQL File 31* | VW_CUSTOMER_FLAT_PAYM...

Limit to 1000 rows

```
## TO VIEW CUSTOMER PAYMENT DETAILS

CREATE VIEW MYDB.VW_CUSTOMER_FLAT_PAYMENT_DETAILS_FOR_ADMIN AS
SELECT CUSTOMER.ID, CUSTOMER.FIRST_NAME, CUSTOMER.LAST_NAME,ORDERS.ORDER_ID,
PAYMENTS.PAYMENT_DATE, PAYMENTS.ACCOUNT_NUMBER, PAYMENTS.PAYMENT_AMOUNT, FLATS.FLAT_ID AS 'FLAT_BOOKED',
PAYMENT_STATUS_MASTER.STATUS_NAME AS 'PAYMENT STATUS'
FROM MYDB.CUSTOMER
INNER JOIN MYDB.ORDERS
ON CUSTOMER.ID = ORDERS.CUSTOMER_ID
INNER JOIN MYDB.PAYMENTS
ON ORDERS.ORDER_ID = PAYMENTS.ORDER_ID
INNER JOIN MYDB.FLATS
ON ORDERS.FLAT_ID = FLATS.FLAT_ID
INNER JOIN MYDB.PAYMENT_STATUS_MASTER
ON PAYMENT_STATUS_MASTER.STATUS_ID = PAYMENTS.STATUS_ID
WITH CHECK OPTION;

# DROP VIEW VW_CUSTOMER_FLAT_PAYMENT_DETAILS_FOR_ADMIN
```

PROJECT_SCRIPT | SQL File 25* | SQL File 31*

Limit to 1000 rows

```
SELECT * FROM MYDB.VW_CUSTOMER_FLAT_PAYMENT_DETAILS_FOR_ADMIN;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	FIRST_NAME	LAST_NAME	ORDER_ID	PAYMENT_DATE	ACCOUNT_NUMBER	PAYMENT_AMOUNT	FLAT_BOOKED	PAYMENT STATUS
1	Shruti	Shetty	5	2017-12-13 04:32:06.000000	67897789468	250	10	PAYMENT VERIFIE
2	Swathi	Shetty	6	2017-12-13 04:32:06.000000	68956885686	100	12	PAYMENT VERIFIE
4	Satisha	Shetty	8	2017-12-13 04:32:07.000000	67856885556	200	16	PAYMENT VERIFIE
4	Satisha	Shetty	13	2017-12-13 04:32:07.000000	67856885556	300	123	PAYMENT VERIFIE
3	Arjun	Shetty	7	2017-12-13 04:33:57.000000	68956885556	150	13	PAYMENT VERIFIE
5	Prema	Shetty	15	2017-12-13 05:16:02.000000	68959985556	250	101	PAYMENT VERIFIE
6	Siddharth	Kotian	16	2017-12-13 06:00:11.000000	68666885556	250	102	PAYMENT VERIFIE
6	Siddharth	Kotian	17	2017-12-13 11:16:45.000000	68666885556	250	5	PAYMENT REJECT

10. Triggers

1. TR_AFTER_AUTHORISE_INSERT

Once a payment is authorised this trigger is executed which generates an invoice Id. It also updates financial Transaction Table with the new Invoice Id. The Status of the payment becomes 20 which means that the payment is verified and authorised. Also, if this payment is the last payment done by the Customer and the Status of the building in which the flat is booked is Completed then on verification the address of the Customer is updated.

Query 1 | SP_AUTHORIZE_PAYMENTS | SQL File 4* | TR_AFTER_AUTHORISE_INSE...

Limit to 1000 rows

```
DROP TRIGGER IF EXISTS MYDB.TR_AFTER_AUTHORISE_INSERT;

DELIMITER $$
CREATE TRIGGER MYDB.TR_AFTER_AUTHORISE_INSERT
    AFTER INSERT ON MYDB.AUTHORIZES
        FOR EACH ROW
        BEGIN
        DECLARE V_ORDER_ID INT;
        DECLARE V_CUSTOMER_ID BIGINT(30);
        DECLARE V_FLAT_ID INT;
        DECLARE V_BALANCE_AMOUNT DOUBLE;
        DECLARE V_BUILDING_ID INT;
        DECLARE V_FLAT_NO INT;
        DECLARE V_STATUS VARCHAR(45);

            #INSERT VALUE IN INVOICES TABLE AFTER AUTHORIZATION
            INSERT INTO MYDB.INVOICES (PAYMENT_ID) VALUES (NEW.PAYMENT_ID);

            #UPDATE INVOICE_ID IN FINANCIAL_TRANSACTION FOR THE PAYMENT AUTHORISED
            UPDATE MYDB.FINANCIAL_TRANSACTION SET INVOICE_ID = LAST_INSERT_ID() WHERE PAYMENT_ID = NEW.PAYMENT_ID;



            SELECT ORDER_ID INTO V_ORDER_ID FROM MYDB.PAYMENTS WHERE PAYMENTS.PAYMENT_ID = NEW.PAYMENT_ID;

            SELECT ORDERS.CUSTOMER_ID, ORDERS.FLAT_ID INTO V_CUSTOMER_ID, V_FLAT_ID FROM MYDB.ORDERS
            WHERE ORDERS.ORDER_ID = V_ORDER_ID;
```

2. TR_AFTER_BUILDING_INSERT

This will create flat record for each floor and allocate the flat details for every floor each time a building is inserted.

```
DROP TRIGGER IF EXISTS MYDB.TR_AFTER_BUILDING_INSERT;
DELIMITER  $$
CREATE TRIGGER  MYDB.TR_AFTER_BUILDING_INSERT
    AFTER INSERT ON MYDB.BUILDINGS
    FOR EACH ROW
BEGIN

    DECLARE TOTAL_FLOORS INT;
    DECLARE ROOM_NO INT;
    DECLARE SELLING_PRICE DOUBLE;
    DECLARE COST_PRICE DOUBLE;
    SET TOTAL_FLOORS = 1;
        WHILE TOTAL_FLOORS <= NEW.NO_OF_FLOORS DO
            SET ROOM_NO = 1;
            WHILE ROOM_NO <= NEW.FLATS_PER_FLOOR DO
                IF TOTAL_FLOORS BETWEEN 1 AND 5 THEN
                    SET COST_PRICE = 100;
                    SET SELLING_PRICE = 300;
                ELSEIF TOTAL_FLOORS BETWEEN 6 AND 10 THEN
                        SET COST_PRICE = 300;
                        SET SELLING_PRICE = 500;
                ELSEIF TOTAL_FLOORS BETWEEN 11 AND 15 THEN
                        SET COST_PRICE = 500;
                        SET SELLING_PRICE = 700;
                ELSEIF TOTAL_FLOORS BETWEEN 16 AND 20 THEN
                        SET COST_PRICE = 700;
                        SET SELLING_PRICE = 900;
```

3. TR_AFTER_ORDERS_INSERT

This trigger updates the order_id in the flats table and updates the flat_booked_status to Y indicating that this flat is booked and hence cannot be booked again.

```
DROP TRIGGER IF EXISTS MYDB.TR_AFTER_ORDERS_INSERT;

DELIMITER $$
CREATE TRIGGER MYDB.TR_AFTER_ORDERS_INSERT
    AFTER INSERT ON MYDB.ORDERS
    FOR EACH ROW
BEGIN
    UPDATE MYDB.FLATS SET ORDER_ID = NEW.ORDER_ID, FLAT_BOOKED_STATUS='Y' WHERE FLAT_ID = NEW.FLAT_ID;
END $$
DELIMITER ;
```

4. TR_BEFORE_BUILDS_INSERT

This trigger is written to ensure that a valid building_id, employee_id and floor_ No is inserted into the builds table.

PROJECT_SCRIPT | SQL File 25* | TR_BEFORE_BUILDS_INSERT

```
DROP TRIGGER MYDB.TR_BEFORE_BUILDS_INSERT;

DELIMITER $$
CREATE TRIGGER MYDB.TR_BEFORE_BUILDS_INSERT
BEFORE INSERT ON BUILDS
FOR EACH ROW
    BEGIN
    DECLARE MSG VARCHAR(100);
    DECLARE V_CONSTRUCTION_STATUS VARCHAR(45);
        IF NEW.BUILDING_ID  NOT IN (SELECT DISTINCT (BUILDING_ID) FROM BUILDINGS) THEN
            SET MSG = 'INVALID BUILDING ID';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = MSG;
            IF NEW.EMPLOYEE_ID NOT IN (SELECT DISTINCT(EMPLOYEE_ID) FROM EMPLOYEES) THEN
                SET MSG = 'INVALID EMPLOYEE ID';
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = MSG;
                IF NEW.FLOORS_COMPLETED > (SELECT NO_OF_FLOORS FROM BUILDINGS WHERE BUILDING_ID = NEW.BUILDING_ID) THE
                    SET MSG = 'INVALID FLOOR NO';
                    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = MSG;
                    #SELECT CONSTRUCTION_STATUS INTO V_CONSTRUCTION_STATUS FROM MYDB.BUILDS WHERE BUILDING_ID
                    #IN (SELECT BUILDING_ID FROM MYDB.BUILDS WHERE BUILDS);
                END IF;
                END IF;
        END IF;

    END $$
DELIMITER ;
```

5. TR_BEFORE_BUILDS_UPDATE

This trigger is written to ensure that a valid building_id, employee_id and floor_ No is updated into the builds table.

PROJECT_SCRIPT | SQL File 25* | TR_BEFORE_BUILDS_INSERT | TR_BEFORE_BUILDS_UPDATE

```
DROP TRIGGER MYDB.TR_BEFORE_BUILDS_INSERT;

DELIMITER $$
CREATE TRIGGER MYDB.TR_BEFORE_BUILDS_UPDATE
BEFORE UPDATE ON BUILDS
FOR EACH ROW
    BEGIN
    DECLARE MSG VARCHAR(100);
    DECLARE V_CONSTRUCTION_STATUS VARCHAR(45);
        IF NEW.BUILDING_ID  NOT IN (SELECT DISTINCT (BUILDING_ID) FROM BUILDINGS) THEN
            SET MSG = 'INVALID BUILDING ID';
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = MSG;
            IF NEW.EMPLOYEE_ID NOT IN (SELECT DISTINCT(EMPLOYEE_ID) FROM EMPLOYEES) THEN
                SET MSG = 'INVALID EMPLOYEE ID';
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = MSG;
                IF NEW.FLOORS_COMPLETED > (SELECT NO_OF_FLOORS FROM BUILDINGS WHERE BUILDING_ID = NEW.BUILDING_ID) TH
                    SET MSG = 'INVALID FLOOR NO';
                    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = MSG;
                    #SELECT CONSTRUCTION_STATUS INTO V_CONSTRUCTION_STATUS FROM MYDB.BUILDS WHERE BUILDING_ID
                    #IN (SELECT BUILDING_ID FROM MYDB.BUILDS WHERE BUILDS);
                END IF;
                END IF;
        END IF;
    END $$
DELIMITER ;
```

11. Stored Procedures

1. SP_AUTHORIZE_PAYMENTS

This procedure is used to insert the verification of payment details into Authorizes tables. One can query the authorizer from this table. Once a Payment is authorised by the employee an Invoice is generated against the payment made.

TR_AFTER_AUTHORISE_INSERT is triggered after insert into authorise table

PROJECT_SCRIPT* | SQL File 25* | SP_MAKE_PAYMENT | SP_AUTHORIZE_PAYMENTS

Limit to 1000 rows

```sql
DROP PROCEDURE IF EXISTS MYDB.SP_AUTHORIZE_PAYMENTS;

DELIMITER $$
CREATE PROCEDURE MYDB.SP_AUTHORIZE_PAYMENTS ( IN AUTHORIZE_PAYMENT_ID INT, IN AUTHORIZER INT, IN PAYMENT_STATUS INT)
BEGIN
    INSERT INTO MYDB.AUTHORIZES(PAYMENT_ID, EMPLOYEE_ID) VALUES (AUTHORIZE_PAYMENT_ID,AUTHORIZER);

     #UPDATE PAYMENT_STATUS IN PAYMENTS TABLE
    UPDATE MYDB.PAYMENTS SET STATUS_ID = PAYMENT_STATUS WHERE PAYMENT_ID = AUTHORIZE_PAYMENT_ID;
END $$
DELIMITER ;
```

```sql
CALL SP_AUTHORIZE_PAYMENTS(24,2,20); -- PAYMENT_ID, EMPLOYEE_ID, PAYMENT_STATUS

SELECT * FROM INVOICES;

SELECT * FROM PAYMENTS; -- PAYMENTS STATUS UPDATED

SELECT * FROM FINANCIAL_TRANSACTION;

-- GET BALANCE AMOUNT
SELECT FN_GET_BALANCE_AMOUNT(6,102); -- CUSTOMER_ID, FLAT_ID

-- PLACE ORDER FOR EQUIPMENTS
SELECT * FROM EQUIPMENTS_HAS_MANUFACTURER;

SELECT * FROM EQUIPMENTS;

CALL SP_PLACE_EQUIPMENT_ORDER(2,1,2); -- EMPLOYEE_ID, QTY, EQUIPMENTS_HAS_MANUFACTURER_ID

SELECT * FROM EQUIPMENTS;
```

Output

Action Output

#	Time	Action	Message
75	17:13:06	SELECT * FROM PAYMENTS WHERE ORDER_ID=19 LIMIT 0, 1000	1 row(s) returned
76	17:14:02	CALL SP_AUTHORIZE_PAYMENTS(24,2,20)	1 row(s) affected

2. SP_CANCEL_ORDER

This procedure is written in case the Customer wants to cancel his flat booking. This will make the order status as “Cancelled” and update the flat booked status. The Customer will not be returned his Initial Payment Amount and hence if only one Payment is made then no money is returned.

```
DROP PROCEDURE IF EXISTS MYDB.SP_CANCEL_ORDER;

DELIMITER $$
    CREATE PROCEDURE MYDB.SP_CANCEL_ORDER (IN NEW_CUSTOMER_ID BIGINT(30), IN FLAT_ID_TOBE_PLACED INT)
        BEGIN
        DECLARE V_ORDER_ID INT;
        DECLARE V_COUNT INT;
         DECLARE RESULT TEXT;
         DECLARE V_PAYMENT_ID INT;
            UPDATE MYDB.ORDERS SET ORDER_STATUS ='Cancelled' WHERE FLAT_ID = FLAT_ID_TOBE_PLACED
            AND CUSTOMER_ID = NEW_CUSTOMER_ID;
            UPDATE MYDB.FLATS SET FLAT_BOOKED_STATUS = 'N' WHERE FLAT_ID = FLAT_ID_TOBE_PLACED;

            SELECT ORDER_ID INTO V_ORDER_ID FROM MYDB.ORDERS WHERE FLAT_ID = FLAT_ID_TOBE_PLACED;

            SELECT COUNT(ORDER_ID) INTO V_COUNT FROM PAYMENTS WHERE ORDER_ID = V_ORDER_ID;

            IF V_COUNT = 0 THEN
                SET RESULT = 'NO ORDER FOUND';
            ELSE
                UPDATE MYDB.ORDERS SET ORDER_STATUS ='Cancelled' WHERE FLAT_ID = FLAT_ID_TOBE_PLACED
                AND CUSTOMER_ID = NEW_CUSTOMER_ID;
                UPDATE MYDB.FLATS SET FLAT_BOOKED_STATUS = 'N' WHERE FLAT_ID = FLAT_ID_TOBE_PLACED;

                INSERT INTO MYDB.DUMMY_PAYMENT (PAYMENT_ID,PAYMENT_DATE,STATUS_ID,ORDER_ID,ACCOUNT_NUMBER,PAYMENT_AI
                SELECT PAYMENT_ID, PAYMENT_DATE,STATUS_ID,ORDER_ID,ACCOUNT_NUMBER,PAYMENT_AMOUNT
                FROM PAYMENTS WHERE ORDER_ID = V_ORDER_ID LIMIT 1;
```

```
SQL File 11   SQL File 21   SQL File 12   TR_AFTER_AUTHORISE_INSE...   SP_CANCEL_ORDER
Limit to 1000 rows
22                  AND CUSTOMER_ID = NEW_CUSTOMER_ID;
23                  UPDATE MYDB.FLATS SET FLAT_BOOKED_STATUS = 'N' WHERE FLAT_ID = FLAT_ID_TOBE_PLACED;
24
25                  INSERT INTO MYDB.DUMMY_PAYMENT (PAYMENT_ID,PAYMENT_DATE,STATUS_ID,ORDER_ID,ACCOUNT_NUMBER,PAYMENT_AI
26                  SELECT PAYMENT_ID, PAYMENT_DATE,STATUS_ID,ORDER_ID,ACCOUNT_NUMBER,PAYMENT_AMOUNT
27                  FROM PAYMENTS WHERE ORDER_ID = V_ORDER_ID LIMIT 1;
28
29                  DELETE FROM MYDB.FINANCIAL_TRANSACTION WHERE PAYMENT_ID IN
30                  (SELECT PAYMENT_ID FROM PAYMENTS WHERE ORDER_ID = V_ORDER_ID);
31
32                  DELETE FROM MYDB.AUTHORIZES WHERE PAYMENT_ID IN
33                  (SELECT PAYMENT_ID FROM PAYMENTS WHERE ORDER_ID = V_ORDER_ID);
34
35                  DELETE FROM MYDB.INVOICES WHERE PAYMENT_ID IN
36                   (SELECT PAYMENT_ID FROM PAYMENTS WHERE ORDER_ID = V_ORDER_ID);
37
38                  DELETE FROM MYDB.PAYMENTS WHERE ORDER_ID = V_ORDER_ID;
39
40                  IF V_COUNT = 1 THEN
41                      SET RESULT = 'ORDER CANCELLED BUT MONEY WILL NOT BE RETURNED';
42                  ELSE
43                      SET RESULT = 'ORDER CANCELLED SUCCESSFULLY';
44                  END IF;
45
46              END IF;
47              SELECT RESULT AS MSG;
48          END $$
49      DELIMITER ;
```

3. SP_GET_TOTAL_PROFIT

This procedure is written to get the total profit of the Construction company.

If this procedure returns a negative value then it means that the Company is in loss else it is in profit

```
DROP PROCEDURE IF EXISTS MYDB.SP_GET_TOTAL_PROFIT;

DELIMITER $$
    CREATE PROCEDURE MYDB.SP_GET_TOTAL_PROFIT (OUT V_TOTAL_PROFIT DOUBLE)
    BEGIN
        DECLARE V_TOTAL_COST_PRICE DOUBLE;
    DECLARE V_TOTAL_SELLING_PRICE DOUBLE;
    DECLARE V_TOTAL_EQUIPMENT_ORDERS_PRICE DOUBLE;


    SELECT SUM(FLAT_COST_PRICE) INTO V_TOTAL_COST_PRICE FROM MYDB.FLATS;

    SELECT SUM(FLAT_SELLING_PRICE) INTO V_TOTAL_SELLING_PRICE FROM MYDB.FLATS
    WHERE FLATS.FLAT_BOOKED_STATUS = 'Y';

    SELECT SUM(TOTAL_PRICE) INTO V_TOTAL_EQUIPMENT_ORDERS_PRICE FROM MYDB.VW_GET_EQUIPMENT_ORDER_DETAILS;

    SET V_TOTAL_PROFIT = V_TOTAL_SELLING_PRICE - (V_TOTAL_COST_PRICE + V_TOTAL_EQUIPMENT_ORDERS_PRICE);

    END $$
DELIMITER ;
```

```
call SP_GET_TOTAL_PROFIT(@total);

select @total;
```

@total
-984800

4. SP_INSERT_INTO_BUILDS

   This procedure enables an employee to build a new building by initiating its construction or updating an existing under constructed building

   TR_BEFORE_BUILDS_INSERT and **TR_BEFORE_BUILDS_UPDATE** are executed to check building_id, employee_id and floor_no before insert and update of the Construction Status.

PROJECT_SCRIPT* | SQL File 25* | SP_MAKE_PAYMENT | SP_AUTHORIZE_PAYMENTS | SP_INSERT_INTO_BUILDS

Limit to 1000 rows

```
DROP PROCEDURE IF EXISTS MYDB.SP_INSERT_INTO_BUILDS;

DELIMITER $$
CREATE PROCEDURE MYDB.SP_INSERT_INTO_BUILDS(IN EMPL_ID INT, IN BLD_ID INT, IN V_FLOOR_COMPLETED INT)
BEGIN
    DECLARE V_TOTAL_FLOORS INT;
    DECLARE V_COUNT INT;

    SELECT NO_OF_FLOORS INTO V_TOTAL_FLOORS FROM MYDB.BUILDINGS  WHERE BUILDING_ID = BLD_ID;
    SELECT COUNT(*) INTO V_COUNT FROM MYDB.BUILDS WHERE BUILDING_ID = BLD_ID;

    IF V_COUNT = 0 THEN
        INSERT INTO MYDB.BUILDS (EMPLOYEE_ID, BUILDING_ID, CONSTRUCTION_STATUS, FLOORS_COMPLETED)
        VALUES (EMPL_ID, BLD_ID, FN_GET_CONSTRUCTION_STATUS(V_FLOOR_COMPLETED,V_TOTAL_FLOORS), V_FLOOR_COMPLETED);
    ELSE
        UPDATE MYDB.BUILDS SET CONSTRUCTION_STATUS = FN_GET_CONSTRUCTION_STATUS(V_FLOOR_COMPLETED,V_TOTAL_FLOORS),
        FLOORS_COMPLETED = V_FLOOR_COMPLETED,BUILDS.EMPLOYEE_ID = EMPL_ID  WHERE BUILDS.BUILDING_ID = BLD_ID;
    END IF;
END $$
DELIMITER ;
```

SP_AUTHORIZE_PAYMENTS | SQL File 4* | SP_INSERT_INTO_BUILDS | TR_BEFORE_BUILDS_INSERT | TR_BEFORE_BUILDS_UPDATE

Limit to 1000 rows

```
CALL SP_INSERT_INTO_BUILDS(1,4,9);
```

Output

Action Output

#	Time	Action	Message
1	00:10:15	CALL SP_INSERT_INTO_BUILDS(1,4,9)	Error Code: 1054. Unknown column 'FLOOR_COMPLETED' in 'field list'
2	00:13:42	CALL SP_INSERT_INTO_BUILDS(1,4,9)	0 row(s) affected

5. SP_MAKE_PAYMENT_BY_CUSTOMER

This procedure is used to make payment for the flat booked by the Customer. This procedure inserts details in the Payments table with the payment details. It also inserts records in Financial_Transaction Table which will keep a track of all payments in the System. It takes input Order_Id, Customer_Account No,Amount to be paid and if its an Online, Cash or Cheque payment

```
DELIMITER $$
CREATE PROCEDURE MYDB.SP_MAKE_PAYMENT_BY_CUSTOMER (IN PLACED_ORDER_ID INT, IN CUSTOMER_ACCOUNT_NO VARCHAR(45), IN A
BEGIN
    DECLARE CODE CHAR(5) DEFAULT '00000';
    DECLARE MSG TEXT;
    DECLARE RESULT TEXT;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
            CODE = RETURNED_SQLSTATE, MSG = MESSAGE_TEXT;
        END;
        IF CUSTOMER_ACCOUNT_NO IS NOT NULL AND PAYMENT_TYPE IN(1,2)  THEN
            IF  CUSTOMER_ACCOUNT_NO IN (SELECT DISTINCT(ACCOUNT_NUMBER) FROM ACCOUNTS WHERE CUSTOMER_ID IN
            (SELECT CUSTOMER_ID FROM ORDERS WHERE ORDER_ID = PLACED_ORDER_ID)) THEN
                IF PAYMENT_TYPE IN (SELECT TRANSACTION_ID FROM MYDB.TRANSACTIONTYPE) THEN
                    ## INSERT VALUES IN PAYMENT TABLE
                    INSERT INTO MYDB.PAYMENTS (PAYMENT_DATE,STATUS_ID,ORDER_ID,ACCOUNT_NUMBER,PAYMENT_AMOUNT)
                    VALUES (NOW(),10,PLACED_ORDER_ID, CUSTOMER_ACCOUNT_NO,AMOUNT_TO_BE_PAID);

                    ## INSERT VALUES IN FINANCIAL_TRANSACTION TABLE
                    INSERT INTO MYDB.FINANCIAL_TRANSACTION (TRANSACTION_ID, PAYMENT_ID)
                    VALUES (PAYMENT_TYPE, LAST_INSERT_ID());

                    IF CODE ='00000' THEN
                        SET RESULT = 'PAYMENT SUCCESSFULL';
                    ELSE
                        SET RESULT = CONCAT('PAYMENT UNSUCCESSFUL, MESSAGE = ',MSG);
                    END IF;
                ELSE
                    SET RESULT = 'INVALID TRANSACTION TYPE';
                END IF;
            ELSE
```

```
SELECT * FROM FLATS;

-- PLACE ORDER FOR A FLAT
CALL MYDB.SP_PLACE_ORDERS(6,103); --  CUSTOMER_ID, FLAT_ID

SELECT * FROM MYDB.ORDERS WHERE FLAT_ID=103; -- TO VIEW ORDERS

SELECT * FROM ACCOUNTS WHERE CUSTOMER_ID=6;

-- MAKE PAYMENT
CALL  SP_MAKE_PAYMENT_BY_CUSTOMER(19,'68666885556',250,1); -- ORDER_ID, ACCOUNT NO, AMOUNT TO BE PAID, TYPE OF PAYM

SELECT * FROM PAYMENTS;
```

RESULT
PAYMENT SUCCESSFULL

6. SP_PLACE_EQUIPMENT_ORDER

The procedure will take input Employee Id, Quantity and the Equipment_Manufacturer from which the Employee wishes to buy. It will check for availability of the Quantity. If the Quantity is available it will update the quantity after subtracting from what is placed. If not it will notify that ordered quantity is unavailable.

```
DELIMITER $$
    CREATE PROCEDURE MYDB.SP_PLACE_EQUIPMENT_ORDER (IN EMP_ID INT, IN QTY INT, IN EQUIP_MANFAC_ID INT)
    BEGIN
        DECLARE V_TOTAL_QTY INT;
        DECLARE V_AVAIL_QTY INT;
        DECLARE V_EQUIP_ID INT;

        SELECT EQUIPMENT_ID INTO V_EQUIP_ID FROM MYDB.EQUIPMENTS_HAS_MANUFACTURER
        WHERE ID = EQUIP_MANFAC_ID;

        SELECT QUANTITY INTO V_TOTAL_QTY FROM MYDB.EQUIPMENTS WHERE EQUIPMENT_ID = V_EQUIP_ID;

        IF V_TOTAL_QTY > QTY THEN

        SET V_AVAIL_QTY = V_TOTAL_QTY - QTY;

        INSERT INTO MYDB.EQUIPMENT_ORDERS (EMPLOYEE_ID, QUANTITY,EQUIPMENT_HAS_MANUFACTURER_ID)
        VALUES (EMP_ID, QTY, EQUIP_MANFAC_ID);

        UPDATE MYDB.EQUIPMENTS SET QUANTITY = V_AVAIL_QTY WHERE EQUIPMENT_ID = V_EQUIP_ID;

        SELECT 'EQUIPMENT ORDER PLACED SUCCESSFULLY' AS MSG;
        ELSE
            SELECT 'QUANTITY UNAVAILABLE' AS MSG;
        END IF;
    END $$
DELIMITER ;
```

```
-- PLACE ORDER FOR EQUIPMENTS
SELECT * FROM EQUIPMENTS_HAS_MANUFACTURER;

SELECT * FROM EQUIPMENTS;

CALL SP_PLACE_EQUIPMENT_ORDER(3,3,3); -- EMPLOYEE_ID, QTY, EQUIPMENTS_HAS_MANUFACTURER_ID

SELECT * FROM EQUIPMENTS;

-- UPDATING THE CONSTRUCTION STATUS

SELECT * FROM BUILDS;
```

MSG
EOUIPMENT ORDER PLACED SUCCESSFULLY

7. SP_PLACE_ORDERS

This procedure is used to book a flat by the Customer. If the flat is already booked the customer gets a notification that the flat is already booked, and it does not allow customer to book the same flat. On Update **TR_AFTER_ORDERS_INSERT** is executed which will update the Order Id and Flat Status booked to Y in Flats table.

```
DROP PROCEDURE IF EXISTS MYDB.SP_PLACE_ORDERS;

DELIMITER $$
    CREATE PROCEDURE MYDB.SP_PLACE_ORDERS (IN NEW_CUSTOMER_ID BIGINT(30), IN FLAT_ID_TOBE_PLACED INT)
    BEGIN
        DECLARE FLAT_STATUS VARCHAR(10);
        SELECT FLAT_BOOKED_STATUS INTO FLAT_STATUS FROM MYDB.FLATS WHERE FLAT_ID = FLAT_ID_TOBE_PLACED;
        IF (FLAT_STATUS ='N') THEN
            INSERT INTO MYDB.ORDERS (CUSTOMER_ID,FLAT_ID,ORDER_STATUS)
            VALUES (NEW_CUSTOMER_ID,FLAT_ID_TOBE_PLACED,'Booked');
            SELECT 'FLAT BOOKED' AS MSG;
        ELSE
            SELECT 'FLAT ALREADY BOOKED' AS MSG;
        END IF ;
    END $$
DELIMITER ;
```

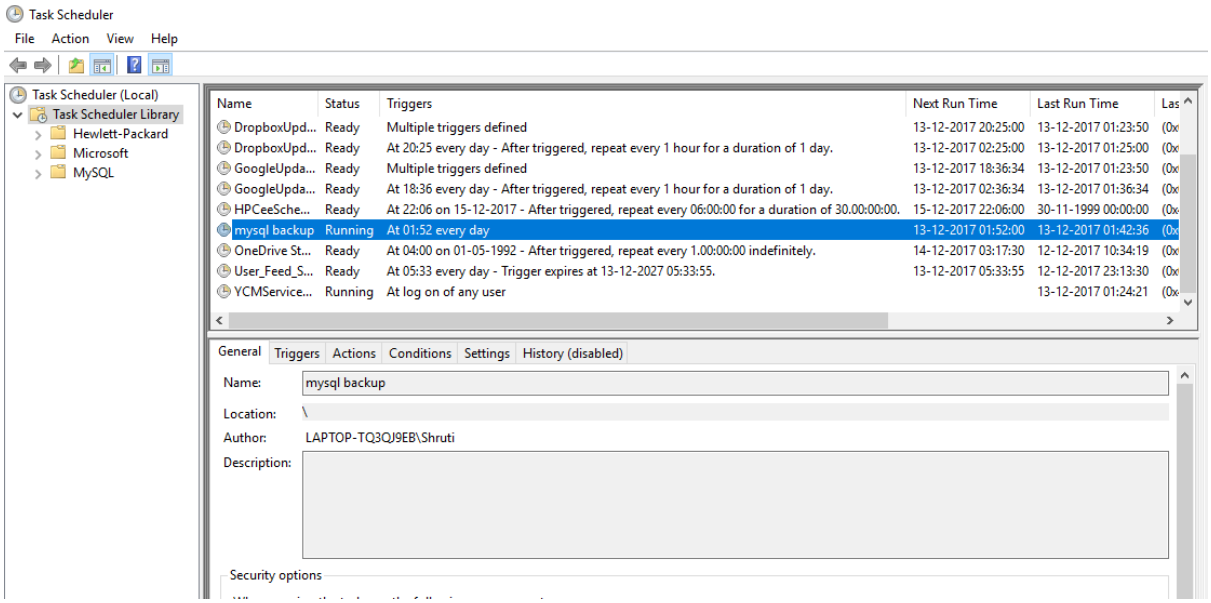
```
USE MYDB;

SELECT * FROM FLATS;

-- PLACE ORDER FOR A FLAT
CALL MYDB.SP_PLACE_ORDERS(6,103); --  CUSTOMER_ID, FLAT_ID
```

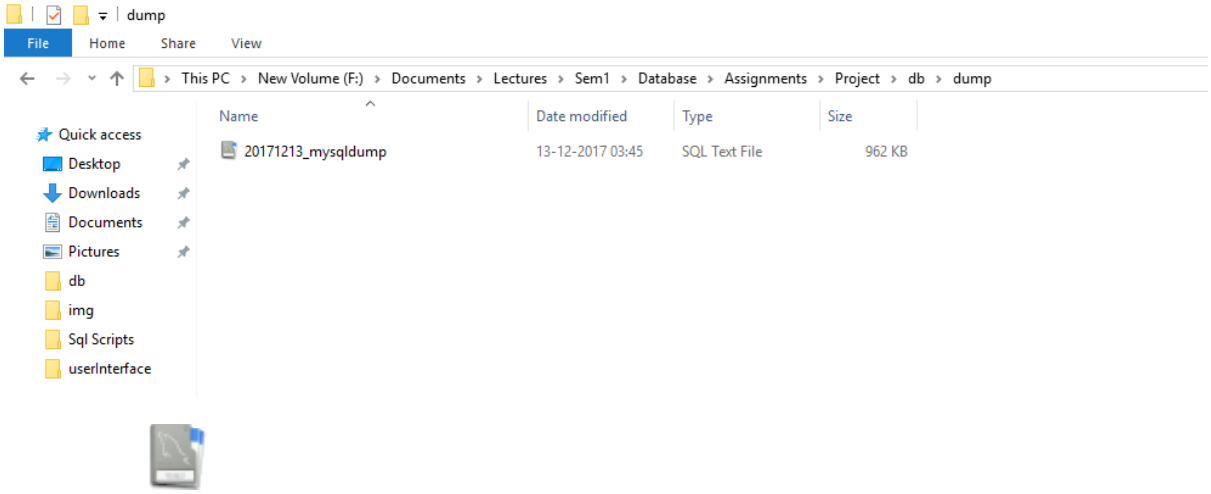
MSG
FLAT BOOKED

12. Backup and Task Schedule

A bat file is written "**mysql-backup.bat**" which will take dump of all the database. This bat file is called by a Windows Task Scheduler which will execute the bat file at the given specific time and will take dump of the entire database. Currently this task is schedule to execute daily at 01:52 AM. Every day a new file will be created along with date appended in the filename.

mysql-backup.bat





20171213_mysqldump.sql