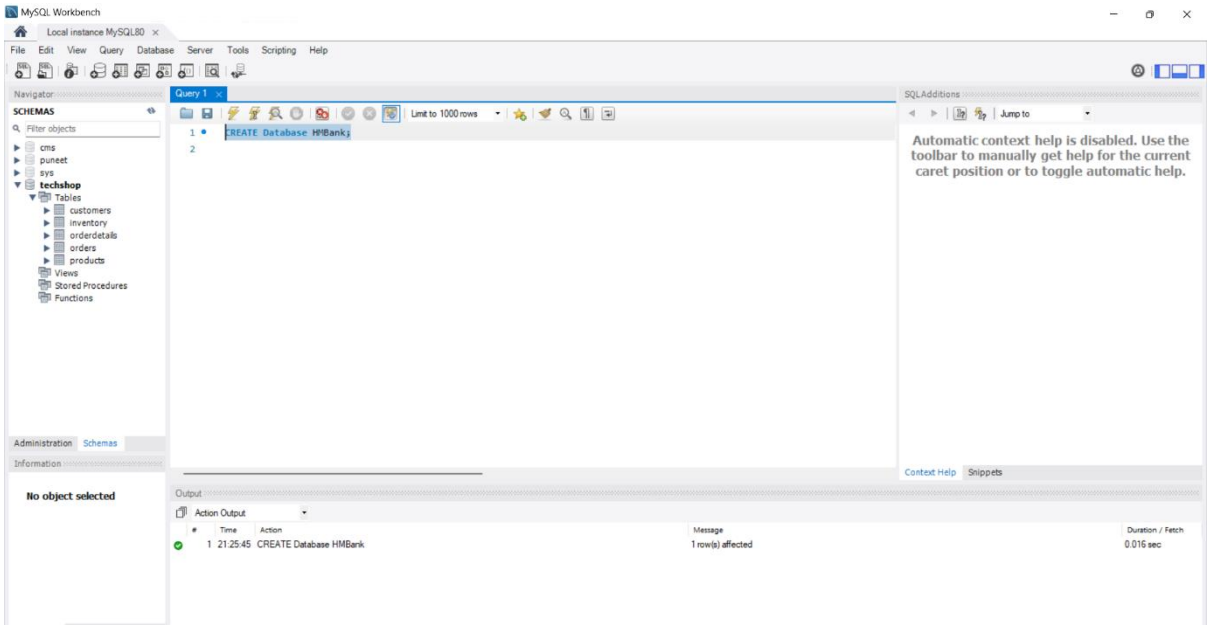
Assignment 3

Task 1:

1. CREATE Database HMBank;



2. Schema for Customer:

```
CREATE table Customers(
customer_id INT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
DOB DATE,
email VARCHAR(50),
phone_number VARCHAR(10)
address VARCHAR(100),
);
```

3. Schema for Accounts:

```
CREATE table Accounts(
account_id INT PRIMARY KEY,
account_type ENUM('savings', 'current', 'zero balance'),
balance DECIMAL (10,2),
FOREIGN KEY(customer_id) REFERENCES Customers(customer_id)
);
```

4. Schema for Transactions:

```
CREATE table Transactions(
transaction_id INT PRIMARY KEY,
transaction_type ENUM('deposit', 'withdrawal', 'transfer'),
amount DECIMAL(10,2),
transaction_date DATETIME,
FOREIGN KEY(account_id) REFERENCES Accounts(account_id)
);
```

5. For customer table:

Primary Key constraint for Customers table:

```
ALTER TABLE Customers
ADD CONSTRAINT P_Customers PRIMARY KEY (customer_id);
```

Unique constraint for email in Customers table:

```
ALTER TABLE Customers
ADD CONSTRAINT U_Email UNIQUE (email);
```

For accounts table:

Primary Key constraint for Accounts table:

```
ALTER TABLE Accounts
ADD CONSTRAINT PK_Accounts PRIMARY KEY (account_id);
```

Foreign Key constraint referencing Customers table:

```
ALTER TABLE Accounts
ADD CONSTRAINT FK_Accounts_Customers FOREIGN KEY (customer_id) REFERENCES
Customers(customer_id);
```

For transactions table:

Primary Key constraint for Transactions table:

```
ALTER TABLE Transactions
ADD CONSTRAINT PK_Transactions PRIMARY KEY (transaction_id);
```

Foreign Key constraint referencing Accounts table:

```
ALTER TABLE Transactions
ADD CONSTRAINT FK_Transactions_Accounts FOREIGN KEY (account_id) REFERENCES
Accounts(account_id);
```

TASK 2:

For Customer table:

```
INSERT INTO Customers (customer_id, first_name, last_name, DOB, email, phone_number,
address)
VALUES
(1, 'Alice', 'Johnson', '1990-03-15', 'alice.johnson@email.com', '1234567890', '123 Main
Street'),
```

(2, 'Bob', 'Smith', '1985-08-22', 'bob.smith@email.com', '9876543210', '456 Oak Avenue'),
(3, 'Charlie', 'Williams', '1992-05-10', 'charlie.williams@email.com', '9812347650', '789 Pine Lane'),
(4, 'David', 'Jones', '1988-11-18', 'david.jones@email.com', '1237890456', '901 Maple Drive'),
(5, 'Eva', 'Brown', '1995-07-03', 'eva.brown@email.com', '2345678901', '232 Cedar Street'),
(6, 'Frank', 'Miller', '1982-09-21', 'frank.miller@email.com', '3456789012', '365 Elm Court'),
(7, 'Grace', 'Davis', '1993-10-12', 'grace.davis@email.com', '4567890123', '567 Birch Road'),
(8, 'Henry', 'Taylor', '1986-06-27', 'henry.taylor@email.com', '5678901234', '666 Willow Lane'),
(9, 'Ivy', 'Moore', '1998-03-07', 'ivy.moore@email.com', '6789012345', '889 Cedar Avenue'),
(10, 'Jack', 'White', '1980-05-05', 'jack.white@email.com', '7890123456', '023 Oak Street');

For Accounts table:

-- INSERT INTO Accounts (account_id, account_type, balance, customer_id)
VALUES
(106, 'zero balance', 0.00, 6),
(102, 'current', 30000.00, 2),
(105, 'current', 60000.00, 5),
(109, 'zero balance', 0.00, 9),
(103, 'zero balance', 0.00, 3),
(107, 'savings', 48000.00, 7),
(101, 'savings', 55000.00, 1),
(110, 'savings', 90000.00, 10),
(108, 'current', 42000.00, 8),
(104, 'savings', 17500.00, 4);

For Transaction table:

-- INSERT INTO Transactions (transaction_id, transaction_type, amount, transaction_date, account_id)
VALUES
(1008, 'withdrawal', 8000.00, '2023-08-15', 108),
(1002, 'withdrawal', 700.00, '2023-02-10', 102),
(1005, 'deposit', 6000.00, '2023-05-25', 105),
(1009, 'deposit', 3000.00, '2023-09-30', 109),
(1007, 'deposit', 1500.00, '2023-07-15', 107),
(1010, 'withdrawal', 1200.00, '2023-10-10', 110),
(1004, 'withdrawal', 4500.00, '2023-04-30', 104),
(1006, 'deposit', 8000.00, '2023-06-25', 106),
(1003, 'deposit', 2000.00, '2023-03-20', 103),
(1001, 'deposit', 1200.00, '2023-01-15', 101);

2.1

```
SELECT c.first_name, c.last_name, a.account_type, c.email
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id;
```

2.2

```
SELECT t.transaction_id, t.transaction_type, t.amount, t.transaction_date
FROM Transactions t
JOIN Accounts a ON t.account_id = a.account_id
JOIN Customers c ON a.customer_id = c.customer_id
WHERE c.customer_id = 1;
```

2.3

```
UPDATE Accounts
SET balance = balance + 1000
WHERE account_id = 103;
```

2.4

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM Customers;
```

2.5

```
DELETE FROM Accounts
WHERE balance = 0 AND account_type = 'savings';
```

2.6

```
SELECT *
FROM Customers
WHERE address LIKE 456 Oak Avenue;
```

2.7

```
SELECT balance
FROM Accounts
WHERE account_id = 108;
```

2.8

```
SELECT *
FROM Accounts
WHERE account_type = 'current' AND balance > 10000.00;
```

2.9

```
SELECT *
FROM Transactions
WHERE account_id = 108;
```

2.10

```
SELECT account_id, balance * 1.5 / 100 AS interest_accrued
FROM Accounts
WHERE account_type = 'savings';
```

2.11

```
SELECT *
FROM Accounts
WHERE balance < 50000;
```

2.12

```
SELECT *
FROM Customers
WHERE address NOT LIKE 456 Oak Avenue;
```

TASK 3:

3.1

```
SELECT AVG(balance) AS average_balance
FROM Accounts;
```

3.2

```
SELECT *
FROM Accounts
ORDER BY balance DESC
LIMIT 10;
```

3.3

```
SELECT SUM(amount) AS total_deposits
FROM Transactions
WHERE transaction_type = 'deposit' AND DATE(transaction_date) = 2023-05-25;
```

3.4

```
SELECT *
FROM Customers
ORDER BY DOB ASC /* or DESC */
LIMIT 1; /*For the oldest*/

SELECT *
FROM Customers
ORDER BY DOB DESC /* or ASC */
LIMIT 1; /* For the newest*/
```

3.5

```
SELECT t.*, a.account_type
FROM Transactions t
JOIN Accounts a ON t.account_id = a.account_id;
```

3.6

```
SELECT c.*, a.*
FROM Customers c
LEFT JOIN Accounts a ON c.customer_id = a.customer_id;
```

3.7

```
SELECT t.*, c.*
FROM Transactions t
JOIN Accounts a ON t.account_id = a.account_id
JOIN Customers c ON a.customer_id = c.customer_id
WHERE t.account_id = 104;
```

3.8

```
SELECT customer_id
FROM Accounts
GROUP BY customer_id
HAVING COUNT(*) > 1;
```

3.9

```
SELECT account_id,
SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE 0 END) -
SUM(CASE WHEN transaction_type = 'withdrawal' THEN amount ELSE 0 END) AS difference
FROM Transactions
GROUP BY account_id;
```

3.10

```
SELECT account_id,
AVG(balance) AS average_daily_balance
FROM Accounts
JOIN Transactions ON Accounts.account_id = Transactions.account_id
WHERE transaction_date BETWEEN 2023-02-13 AND 2023-10-30
GROUP BY account_id;
```

3.11

```
SELECT account_type, SUM(balance) AS total_balance
FROM Accounts
GROUP BY account_type;
```

3.12

```
SELECT account_id, COUNT(*) AS transaction_count
FROM Transactions
GROUP BY account_id
ORDER BY transaction_count DESC;
```

3.13

```
SELECT c.customer_id, c.first_name, c.last_name, a.account_type, SUM(a.balance) AS
total_balance
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
```

```
GROUP BY c.customer_id, a.account_type
HAVING total_balance > 30000;
```

3.14

```
SELECT transaction_id, amount, transaction_date, account_id, COUNT(*) AS occurrences
FROM Transactions
GROUP BY amount, transaction_date, account_id
HAVING occurrences > 1;
```

TASK 4:

4.1

```
SELECT c.*
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
ORDER BY a.balance DESC
LIMIT 1;
```

4.2

```
SELECT customer_id, AVG(balance) AS avg_balance
FROM Accounts
GROUP BY customer_id
HAVING COUNT(*) > 1;
```

4.3

```
SELECT DISTINCT a.*
FROM Accounts a
JOIN Transactions t ON a.account_id = t.account_id
WHERE t.amount > (SELECT AVG(amount) FROM Transactions);
```

4.4

```
SELECT c.*
FROM Customers c
LEFT JOIN Accounts a ON c.customer_id = a.customer_id
LEFT JOIN Transactions t ON a.account_id = t.account_id
WHERE t.transaction_id IS NULL;
```

4.5

```
SELECT SUM(balance) AS total_balance_no_transactions
FROM Accounts
WHERE account_id NOT IN (SELECT account_id FROM Transactions);
```

4.6

```
SELECT t.*
FROM Transactions t
JOIN Accounts a ON t.account_id = a.account_id
WHERE a.balance = (SELECT MIN(balance) FROM Accounts);
```

4.7

```
SELECT customer_id
FROM Accounts
GROUP BY customer_id
HAVING COUNT(DISTINCT account_type) > 1;
```

4.8

```
SELECT account_type, COUNT(*) * 100.0 / (SELECT COUNT(*) FROM Accounts) AS percentage
FROM Accounts
GROUP BY account_type;
```

4.9

```
SELECT t.*
FROM Transactions t
JOIN Accounts a ON t.account_id = a.account_id
WHERE a.customer_id = 9;
```

4.10

```
SELECT account_type,
 (SELECT SUM(balance) FROM Accounts a WHERE a.account_type = acc.account_type) AS
total_balance
FROM (SELECT DISTINCT account_type FROM Accounts) AS acc;
```