BT:

1. -- Create a table named students
CREATE TABLE students (
    -- Student ID, set as primary key, automatically incremented
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    -- Student name, string type, maximum length is 50, cannot be empty
    student_name VARCHAR(50) NOT NULL,
    -- Student age, integer type
    student_age INT,
    -- Student gender, string type, length 1 (e.g. 'M' for male, 'F' for female)
    student_gender CHAR(1),
    -- The student's class, the string type, and the maximum length is 20
    student_class VARCHAR(20)
);

3.

Insert data from the FACULTY table:
INSERT INTO FACULTY (facultyId, facultyName, NoOfStaff) VALUES
('C001', 'Computing', 120),
('E002', 'Engineering', 76),
('M002', 'Mathematics', 56),
('B001', 'Business', 89),
('FG3124', 'Lucy Liu', 1997);
Insert data from the STAFF table:
INSERT INTO STAFF (staffId, staffName, staffDOB, staffFaculty) VALUES
('AB9872', 'Mark White', '01 - JAN - 1978', 'M002'),
('DL2314', 'Jas Singh', '14 - MAR - 1982', 'M002'),
('AF4512', 'Alison Green', '23 - DEC - 1998', 'C001'),
('BK2134', 'Kieran West', '16 - JAN - 1992', 'B001'),
('FG3124', 'Lucy Liu', '03 - AUG - 1997', 'E002');
Reason: The data of the FACULTY table should be inserted first. Because there is a foreign key column staffFaculty in the STAFF table, it refers to the facultyId in the FACULTY table. If the data of the STAFF table is inserted first, the insertion may fail or the foreign key constraint may be violated because there is no corresponding facultyId in the FACULTY table.
When the data of the FACULTY table is inserted, the data of the STAFF table is inserted again, which ensures that the values of the staffFaculty column in the STAFF table have corresponding facultyIDs in the FACULTY table.

4.

a. SELECT * FROM STAFF;
b. SELECT facultyName FROM FACULTY WHERE NoOfStaff < 75;
c. SELECT * FROM STAFF WHERE staffDOB BETWEEN '01 - JAN - 1980' AND '31 - DEC - 1989';
d. SELECT staffId AS ID, staffName AS Name, staffDOB AS 'Date of Birth', staffFaculty AS 'Faculty ID' FROM STAFF ORDER BY staffName DESC;

e. UPDATE STAFF SET staffFaculty = 'E002' WHERE staffName = 'Alison Green';

f. DELETE FROM STAFF WHERE staffName = 'Kieran West';

MT:

5.

1. **ACCOUNT table**

```
CREATE TABLE ACCOUNT (
    account_id INT PRIMARY KEY,
    customer_id INT,
    product_type_id INT,
    balance DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (product_type_id) REFERENCES PRODUCT_TYPE(product_type_id)
);
```

2. **ACC_TRANSACTION table**

```
CREATE TABLE ACC_TRANSACTION (
    transaction_id INT PRIMARY KEY,
    account_id INT,
    transaction_date DATE,
    amount DECIMAL(10,2),
    transaction_type VARCHAR(50),
    FOREIGN KEY (account_id) REFERENCES ACCOUNT(account_id)
);
```

3. **BRANCH table**

```
CREATE TABLE BRANCH (
    branch_id INT PRIMARY KEY,
    branch_name VARCHAR(100),
    address VARCHAR(200),
    phone_number VARCHAR(20)
);
```

4. **BUSINESS table**

```
CREATE TABLE BUSINESS (
    business_id INT PRIMARY KEY,
    business_name VARCHAR(100),
    contact_person VARCHAR(100),
    phone_number VARCHAR(20)
);
```

5. **CUSTOMER table**

```
CREATE TABLE CUSTOMER (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(100),
    address VARCHAR(200),
    phone_number VARCHAR(20),
    is_business BOOLEAN
```

);

6. **DEPARTMENT table**

```
CREATE TABLE DEPARTMENT (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(100)
);
```

7. **EMPLOYEE table**

```
CREATE TABLE EMPLOYEE (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100),
    department_id INT,
    branch_id INT,
    FOREIGN KEY (department_id) REFERENCES DEPARTMENT(department_id),
    FOREIGN KEY (branch_id) REFERENCES BRANCH(branch_id)
);
```

8. **OFFICER table**

```
CREATE TABLE OFFICER (
    officer_id INT PRIMARY KEY,
    company_id INT,
    officer_name VARCHAR(100),
    FOREIGN KEY (company_id) REFERENCES BUSINESS(business_id)
);
```

9. **PRODUCT table**

```
CREATE TABLE PRODUCT (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    product_type_id INT,
    FOREIGN KEY (product_type_id) REFERENCES PRODUCT_TYPE(product_type_id)
);
```

10. **PRODUCT_TYPE table**

```
CREATE TABLE PRODUCT_TYPE (
    product_type_id INT PRIMARY KEY,
    product_type_name VARCHAR(100)
);
```

6.

a) SELECT 28964 * 1.185 AS Increased_Value;

b) SELECT First_Name, Last_Name FROM Employee;

c) SELECT DISTINCT Product_Type FROM Product;

d) SELECT Name FROM Product_Type WHERE Name LIKE '%Loan%';

e) SELECT * FROM Employee WHERE First_Name LIKE 'S%';

f) SELECT * FROM Employee
WHERE (First_Name LIKE 'S%' OR First_Name LIKE 'T%')
AND Dept_ID = (SELECT Dept_ID FROM Department WHERE Name = 'Operations');

g) SELECT Emp_ID, First_Name, Last_Name FROM Employee

```
WHERE First_Name IN ('Susan', 'Helen', 'Paula');
h) SELECT * FROM Employee
WHERE Start_Date > '2001-01-01' AND Start_Date < '2002-12-31';
i) SELECT * FROM Customer WHERE FED_ID LIKE '___-__-____';
j) SELECT Product_Type_CD, Name FROM Product
ORDER BY Product_Type_CD ASC, Name DESC;
k) SELECT * FROM Employee
WHERE Title LIKE '%Teller%'
ORDER BY Start_Date;
l) UPDATE Accounts
SET Available_Balance = Available_Balance * 1.02,
      Pending_Balance = Pending_Balance * 1.02
WHERE Cust_ID = 1;
SELECT Account_ID, Product_CD, Available_Balance, Pending_Balance
FROM Accounts WHERE Cust_ID = 1;
n) SELECT Account_ID, Cust_ID, Available_Balance
FROM Accounts
WHERE Available_Balance > 10000
ORDER BY Available_Balance DESC;
o) SELECT DISTINCT City
FROM Customer
WHERE State = 'NH'
ORDER BY City;
p) UPDATE Customer
SET Last_Name = 'Brown'
WHERE First_Name = 'Susan' AND Last_Name = 'Tingley';
q) SELECT * FROM Individual
WHERE Birth_Date < '1965-01-01';
r) UPDATE Employee
SET End_Date = '2019-11-01'
WHERE First_Name = 'Thomas' AND Last_Name = 'Ziegler';
```