Study if you wanna pass nigga

This is a comprehensive set of SQL exercises covering Data Definition Language (DDL) and Data Manipulation Language (DML) commands, as well as basic querying with grouping, aggregation, and conditional logic.
Since you haven't provided the specific table structures (attributes), I will define standard, typical attributes for an Employee, Department, Student, and Customer table and use a generic structure for the Stock table to answer your questions.
I will use standard SQL syntax, which is generally compatible with major relational database systems like MySQL, PostgreSQL, and SQL Server.
1. Employee Table
I will define the Employee table with common attributes: Emp_ID, E_Name, Job, and Sal.
Table Structure & Creation

```
-- 1. Create the base Employee table
CREATE TABLE Employee (
    Emp_ID INT PRIMARY KEY,
    E_Name VARCHAR(100) NOT NULL,
    Job VARCHAR(50),
    Sal DECIMAL(10, 2)
);
```

1.a. Add Column DoJ

```
-- Add a column DoJ (Date of Joining)
ALTER TABLE Employee
ADD DoJ DATE;
```

1.b. Insert Five Records

```
-- Insert five sample records
INSERT INTO Employee (Emp_ID, E_Name, Job, Sal, DoJ) VALUES
(101, 'Alice Johnson', 'Manager', 60000.00, '2020-05-15'),
(102, 'Bob Williams', 'Clerk', 35000.00, '2023-01-20'),
(103, 'Charlie Brown', 'Analyst', 55000.00, '2021-11-01'),
(104, 'David Lee', 'Salesman', 45000.00, '2022-07-25'),
(105, 'Eva Davis', 'Manager', 62000.00, '2019-03-10');
```

1.c. Update Column Details of Job
This assumes you want to change the Job title for a specific employee, say Emp_ID = 104.

```
-- Update the Job for employee David Lee (Emp_ID 104) from 'Salesman' to 'Senior
Salesman'
UPDATE Employee
SET Job = 'Senior Salesman'
WHERE Emp_ID = 104;
```

1.d. Rename Column Sal as Emp_Salary

```
-- Rename the column Sal to Emp_Salary
ALTER TABLE Employee
RENAME COLUMN Sal TO Emp_Salary; -- Syntax for PostgreSQL/Oracle
```

```
-- ALTER TABLE Employee CHANGE Sal Emp_Salary DECIMAL(10, 2); -- Syntax for
MySQL
-- EXEC sp_rename 'Employee.Sal', 'Emp_Salary', 'COLUMN'; -- Syntax for SQL Server
```

Note: The RENAME COLUMN syntax can vary between SQL dialects.

1.e. Delete an Employee

This deletes the record for the employee with Emp_ID = 102.

```
-- Delete the employee with Emp_ID 102
DELETE FROM Employee
WHERE Emp_ID = 102;
```

2. Department Table

I will define the Department table with Dept_No, D_Name, and Location. I'll also add a Dept_No column to the Employee table for joins, as it's required for part 2.c.

Table Structure & Creation

Assuming the Employee table now has a Dept_No column.

```
-- Update Employee table to add Dept_No for part 2.c
ALTER TABLE Employee
ADD Dept_No INT;

-- 2. Create the Department table
CREATE TABLE Department (
    Dept_No INT PRIMARY KEY,
    D_Name VARCHAR(100) NOT NULL,
    Location VARCHAR(100)
);
```

2.a. Add Column Designation

```
-- Add column designation to the Department table
ALTER TABLE Department
ADD Designation VARCHAR(50);
```

2.b. Insert Values

```
-- Insert values into Department table
INSERT INTO Department (Dept_No, D_Name, Location, Designation) VALUES
(10, 'Accounting', 'New York', 'Senior Accountant'),
(20, 'Research', 'Dallas', 'Lead Researcher'),
(30, 'Sales', 'Chicago', 'Sales Director'),
(40, 'Operations', 'Boston', 'Operations Head');

-- Update Employee records with Dept_No for the query later
UPDATE Employee SET Dept_No = 10 WHERE Emp_ID IN (101, 105);
UPDATE Employee SET Dept_No = 20 WHERE Emp_ID IN (103);
UPDATE Employee SET Dept_No = 30 WHERE Emp_ID IN (104);
-- Note: Employee 102 was deleted in 1.e
```

2.c. List Records of Employee Table Grouped by Dept_No

This will count the number of employees in each department.

```
-- List the number of records (employees) grouped by Dept_No
SELECT
    Dept_No,
    COUNT(Emp_ID) AS Number_of_Employees
FROM
    Employee
GROUP BY
    Dept_No
ORDER BY
    Dept_No;
```

2.d. Update the Record Where Dept_No is 5

Since Dept_No = 5 doesn't exist yet, this query will insert a new department if your SQL dialect supports UPSERT (like MERGE or INSERT... ON CONFLICT UPDATE), or just update an existing one. I'll show a simple UPDATE assuming Dept_No = 20 is the target, as 5 is non-existent.

```
-- Update the Location for Department 20 (assuming '5' was a typo, or for demonstration)
UPDATE Department
SET Location = 'San Francisco'
WHERE Dept_No = 20;
```

2.e. Delete any Column Data from the Table

This action typically means setting the value of a column to NULL for all rows, or dropping the entire column. I'll drop the Designation column as it was just added in part 2.a.

```
-- Delete (Drop) the Designation column from the Department table
ALTER TABLE Department
DROP COLUMN Designation;
```

3. Student Database

I will define the Student table with S_ID, S_Name, Admission_Year, and five subject marks (Sub1 through Sub5).

Table Structure & Creation

```
CREATE TABLE Student (
    S_ID INT PRIMARY KEY,
    S_Name VARCHAR(100) NOT NULL,
    Admission_Year INT,
    Sub1 INT,
    Sub2 INT,
    Sub3 INT,
    Sub4 INT,
    Sub5 INT
);
```

3.a. Insertion

```
-- Insert sample records
INSERT INTO Student (S_ID, S_Name, Admission_Year, Sub1, Sub2, Sub3, Sub4, Sub5)
VALUES
(501, 'Alex Roe', 2022, 85, 78, 62, 50, 91), -- Admitted in range (2022)
```

```
(502, 'Beth Fox', 2023, 40, 65, 75, 80, 55), -- Admitted in range (2023), Fails Sub1
(503, 'Cami Hill', 2021, 95, 88, 92, 90, 98), -- Not in range
(504, 'Dale Ivy', 2022, 60, 45, 50, 70, 80); -- Admitted in range (2022), Fails Sub2, Sub3
```

3.b. Deletion
Delete the student who was not admitted in the 2022-2023 range.

```
-- Delete the record for student Cami Hill (S_ID 503)
DELETE FROM Student
WHERE S_ID = 503;
```

3.c. Generate a Simple Report for Students Admitted Between 2022 to 2023

```
-- List students admitted between (and including) 2022 and 2023
SELECT
    S_ID,
    S_Name,
    Admission_Year,
    (Sub1 + Sub2 + Sub3 + Sub4 + Sub5) AS Total_Marks
FROM
    Student
WHERE
    Admission_Year BETWEEN 2022 AND 2023
ORDER BY
    Admission_Year, S_Name;
```

3.d. Query for Failed Students, Max Mark in Sub1, and Average of Sub2
Assuming a passing mark is 50 in each subject.

```
SELECT
    -- Part 1: List all students who failed in a particular subject (e.g., Sub1)
    GROUP_CONCAT(CASE WHEN Sub1 < 50 THEN S_Name ELSE NULL END) AS
Failed_Sub1_Students,

    -- Part 2: Find the maximum marks in Sub1
    MAX(Sub1) AS Max_Marks_Sub1,

    -- Part 3: Find the average of Sub2 from total 5 subjects
    AVG(Sub2) AS Average_Marks_Sub2
FROM
    Student;
```

Note: GROUP_CONCAT is a MySQL function. For other databases, you might use STRING_AGG (PostgreSQL/SQL Server) or LISTAGG (Oracle).
4. Develop "Employee" Details Table
This section reuses the Employee table created in section 1 (as modified/populated), which now has Emp_ID, E_Name, Job, Emp_Salary, DoJ, and Dept_No.
4.a. Count Employees in Department 20

```
-- Count employees in department 20
SELECT
    COUNT(Emp_ID) AS Employees_in_Dept_20
```

```
FROM
    Employee
WHERE
    Dept_No = 20;
```

4.b. Find the Minimum Salary Earned by Clerk

```
-- Find the minimum salary for employees with the job title 'Clerk'
SELECT
    MIN(Emp_Salary) AS Min_Salary_Clerk
FROM
    Employee
WHERE
    Job = 'Clerk';
```

4.c. Find Minimum, Maximum, Average Salary of All Employees

```
-- Aggregate salaries for all employees
SELECT
    MIN(Emp_Salary) AS Minimum_Salary,
    MAX(Emp_Salary) AS Maximum_Salary,
    AVG(Emp_Salary) AS Average_Salary
FROM
    Employee;
```

4.d. List the Minimum and Maximum Salaries for Each Job Type

```
-- Group by job and find the min/max salary within each group
SELECT
    Job,
    MIN(Emp_Salary) AS Min_Salary_by_Job,
    MAX(Emp_Salary) AS Max_Salary_by_Job
FROM
    Employee
GROUP BY
    Job
ORDER BY
    Job;
```

4.e. List the Employee Names in Descending Order

```
-- Order by employee name (E_Name) in descending order (Z to A)
SELECT
    E_Name
FROM
    Employee
ORDER BY
    E_Name DESC;
```

4.f. List the Employee ID, Names in Ascending Order by Emp_ID

```
-- Order by employee ID (Emp_ID) in ascending order (1 to N)
SELECT
```

```
    Emp_ID,
    E_Name
FROM
    Employee
ORDER BY
    Emp_ID ASC;
```

5. Stock Table
I will define the Stock table with Stock_ID, Item_Name, Price, and Rating.
Table Structure & Creation

```
CREATE TABLE Stock (
    Stock_ID INT PRIMARY KEY,
    Item_Name VARCHAR(100) NOT NULL,
    Price DECIMAL(10, 2),
    Rating INT
);
```

5.a. Add Column Age

```
-- Add a column age (Age of the stock item in the inventory, in days)
ALTER TABLE Stock
ADD Age INT;
```

5.b. Insert Values

```
INSERT INTO Stock (Stock_ID, Item_Name, Price, Rating, Age) VALUES
(201, 'Laptop Pro', 1200.00, 9, 45),
(202, 'Mechanical Keyboard', 150.00, 7, 120),
(203, 'Gaming Mouse', 75.00, 9, 30),
(204, 'Monitor 4K', 450.00, 8, 200),
(205, 'Webcam HD', 50.00, 6, 15);
```

5.c. Delete the Row with Rating >8

```
-- Delete all rows where the Rating is greater than 8
DELETE FROM Stock
WHERE Rating > 8;
```

5.d. Update the Column Details of Stock
This will update the Price for the item 'Monitor 4K'.

```
-- Update the Price of 'Monitor 4K'
UPDATE Stock
SET Price = 420.00 -- Discounted price
WHERE Item_Name = 'Monitor 4K';
```

5.e. Insert Null Values into the Table
This inserts a new record where the Price and Age are unknown (NULL).

```
INSERT INTO Stock (Stock_ID, Item_Name, Price, Rating, Age) VALUES
(206, 'Wireless Headset', NULL, 7, NULL);
```

6. Customer Table

I will define the Customer table with Cust_ID, C_Name, and City.
Table Structure & Creation

```
CREATE TABLE Customer (
    Cust_ID INT PRIMARY KEY,
    C_Name VARCHAR(100) NOT NULL,
    City VARCHAR(50)
);
```

6.a. Insert Records

```
INSERT INTO Customer (Cust_ID, C_Name, City) VALUES
(301, 'Fiona Green', 'London'),
(302, 'George Hall', 'Paris'),
(303, 'Hannah Ivy', 'London'),
(304, 'Ian Jones', 'Berlin');
```

6.b. Add Salary Column

```
-- Add a column Salary to the Customer table
ALTER TABLE Customer
ADD Salary DECIMAL(10, 2);
```

6.c. Alter the Table Column Domain
Change the data type (domain) of the City column to support a larger string.

```
-- Alter the column City to have a larger character limit (e.g., from VARCHAR(50) to
VARCHAR(100))
ALTER TABLE Customer
ALTER COLUMN City TYPE VARCHAR(100); -- Syntax for PostgreSQL
-- ALTER TABLE Customer MODIFY COLUMN City VARCHAR(100); -- Syntax for MySQL
-- ALTER TABLE Customer ALTER COLUMN City VARCHAR(100); -- Syntax for SQL Server
```

6.d. Drop Salary Column

```
-- Drop the Salary column
ALTER TABLE Customer
DROP COLUMN Salary;
```

6.e. Delete the Rows of Customer Table with Condition
Delete all customers from 'Paris'.

```
-- Delete rows where the City is 'Paris'
DELETE FROM Customer
WHERE City = 'Paris';
```

7. Create Two Tables (Employee_Personal_details & Employee_Salary_details)
Table Structure & Creation with Constraints (DDL)

```
-- Parent Table Creation
CREATE TABLE Employee_Personal_details (
    Employee_ID INT PRIMARY KEY NOT NULL UNIQUE, -- PRIMARY KEY implies NOT
NULL and UNIQUE
    P_Name VARCHAR(100) NOT NULL,
    Age INT NOT NULL,
```

```
    Location VARCHAR(100)
);

-- Child Table Creation with Foreign Key
CREATE TABLE Employee_Salary_details (
    Salary_ID INT PRIMARY KEY,
    Employee_ID INT NOT NULL,
    Department_ID INT, -- Added for parts (b) and (e)
    Annual_Salary DECIMAL(10, 2),

    -- Foreign Key Constraint
    FOREIGN KEY (Employee_ID) REFERENCES
Employee_Personal_details(Employee_ID)
);

Insert Sample Data
INSERT INTO Employee_Personal_details (Employee_ID, P_Name, Age, Location)
VALUES
(1, 'Amy Chen', 30, 'Tokyo'),
(2, 'Ben Smith', 45, 'London'),
(3, 'Chris Lee', 28, 'Tokyo'),
(4, 'Dana Roy', 35, 'Paris'),
(5, 'Elias Vaz', 50, 'London');

INSERT INTO Employee_Salary_details (Salary_ID, Employee_ID, Department_ID,
Annual_Salary) VALUES
(1001, 1, 10, 25000.00), -- Tokyo, Dept 10
(1002, 2, 20, 32000.00), -- London, Dept 20
(1003, 3, 10, 25000.00), -- Tokyo, Dept 10
(1004, 4, 30, 19000.00), -- Paris, Dept 30 (Lowest salary in Dept 30)
(1005, 5, 20, 30000.00); -- London, Dept 20

I will assume a minimum salary for Department 30 is 19000.
7.a. Determine the names of employees, who earn more than 20000
SELECT
    P.P_Name
FROM
    Employee_Personal_details P
JOIN
    Employee_Salary_details S ON P.Employee_ID = S.Employee_ID
WHERE
    S.Annual_Salary > 20000.00;

7.b. Determine the names of employees, who take highest salary in their departments
SELECT
    P.P_Name
FROM
    Employee_Personal_details P
```

```
JOIN
   Employee_Salary_details S ON P.Employee_ID = S.Employee_ID
WHERE
   S.Annual_Salary IN (
      SELECT MAX(Annual_Salary)
      FROM Employee_Salary_details
      GROUP BY Department_ID
   );
```

7.c. Determine the employees, who are located at the same place
This query finds pairs of employees who share the same location, excluding comparing an employee with themselves.

```
SELECT
   A.P_Name AS Employee_1,
   B.P_Name AS Employee_2,
   A.Location
FROM
   Employee_Personal_details A
JOIN
   Employee_Personal_details B
   ON A.Location = B.Location AND A.Employee_ID < B.Employee_ID
ORDER BY
   A.Location;
```

7.d. Determine the employees, whose total salary is like the minimum salary of any department
This finds employees whose individual salary matches the minimum salary paid in any department.

```
SELECT
   P.P_Name,
   S.Annual_Salary
FROM
   Employee_Personal_details P
JOIN
   Employee_Salary_details S ON P.Employee_ID = S.Employee_ID
WHERE
   S.Annual_Salary IN (
      SELECT MIN(Annual_Salary)
      FROM Employee_Salary_details
      GROUP BY Department_ID
   );
```

7.e. Determine the department which does not contain any employees
This requires a list of all departments, so I will temporarily create a Department lookup table for this query.

```
-- Temporary Department Table (for lookup)
CREATE TABLE Departments_Lookup (
   Dept_ID INT PRIMARY KEY,
```

```
    Dept_Name VARCHAR(50)
);

INSERT INTO Departments_Lookup (Dept_ID, Dept_Name) VALUES
(10, 'HR'),
(20, 'IT'),
(30, 'Marketing'),
(40, 'Finance'); -- This department has no employees in Employee_Salary_details

-- Query to find departments with no employees
SELECT
    D.Dept_Name
FROM
    Departments_Lookup D
LEFT JOIN
    Employee_Salary_details S ON D.Dept_ID = S.Department_ID
WHERE
    S.Employee_ID IS NULL;
```