The National University of Computer and Emerging Sciences

Assignment 02 & 03:
Database Systems

Sr no.	Name:	Roll no:	Section:
1.	Hasan Yahya	22L-7971	BSE-4B

Question - 01:

Firstly create the Database and add data using these SQL Queries:

```
use master;
GO
CREATE DATABASE Supply_Chain_System;
GO
use [Supply_Chain_System];

-- Creating tables
CREATE TABLE Suppliers
(
snum CHAR(2) PRIMARY KEY NOT NULL,
sname VARCHAR(100) NOT NULL,
status INT NOT NULL,
city VARCHAR(100) NOT NULL
);

CREATE TABLE Parts
(
pnum CHAR(2) PRIMARY KEY NOT NULL,
pname VARCHAR(100) NOT NULL,
color VARCHAR(100) NOT NULL,
weight INT NOT NULL DEFAULT 0,
city VARCHAR(100) NOT NULL
);

CREATE TABLE Jobs
(
jnum CHAR(2) PRIMARY KEY NOT NULL,
jname VARCHAR(100) NOT NULL,
numworkers INT NOT NULL DEFAULT 0,
city VARCHAR(100) NOT NULL
);

CREATE TABLE Shipments
(
snum CHAR(2) NOT NULL,
pnum CHAR(2) NOT NULL,
jnum CHAR(2) NOT NULL,
quantity INT NOT NULL DEFAULT 0,
PRIMARY KEY (snum, pnum, jnum),
FOREIGN KEY (snum) REFERENCES Suppliers(snum),
FOREIGN KEY (pnum) REFERENCES Parts(pnum),
FOREIGN KEY (jnum) REFERENCES Jobs(jnum)
);
```

```sql
-- Inserting data into tables
INSERT INTO Suppliers (snum, sname, status, city)
VALUES ('S1', 'SMITH', 20, 'LONDON'),
('S2', 'JONES', 10, 'PARIS'),
('S3', 'BLAKE', 30, 'PARIS'),
('S4', 'CLARK', 20, 'LONDON'),
('S5', 'ADAMS', 30, 'ATHENS');

INSERT INTO Parts (pnum, pname, color, weight, city)
VALUES ('P1', 'NUT', 'RED', 12, 'LONDON'),
('P2', 'BOLT', 'GREEN', 17, 'PARIS'),
('P3', 'SCREW', 'BLUE', 17, 'ROME'),
('P4', 'SCREW', 'RED', 14, 'LONDON'),
('P5', 'CAM', 'BLUE', 12, 'PARIS'),
('P6', 'COG', 'RED', 19, 'LONDON');

INSERT INTO Jobs (jnum, jname, numworkers, city)
VALUES ('J1', 'SORTER', 20, 'PARIS'),
('J2', 'PUNCH', 10, 'ROME'),
('J3', 'READER', 30, 'ATHENS'),
('J4', 'CONSOLE', 20, 'ATHENS'),
('J5', 'COLLATOR', 30, 'LONDON'),
('J6', 'TERMINAL', 20, 'OSLO'),
('J7', 'TAPE', 10, 'LONDON');

INSERT INTO Shipments (snum, pnum, jnum, quantity)
VALUES ('S1', 'P1', 'J1', 200),
('S1', 'P1', 'J4', 700),
('S2', 'P3', 'J1', 400),
('S2', 'P3', 'J2', 200),
('S2', 'P3', 'J3', 200),
('S2', 'P3', 'J4', 500),
('S2', 'P3', 'J5', 600),
('S2', 'P3', 'J6', 400),
('S2', 'P3', 'J7', 800),
('S2', 'P5', 'J2', 100),
('S3', 'P3', 'J1', 200),
('S3', 'P4', 'J2', 500),
('S4', 'P6', 'J3', 300),
('S4', 'P6', 'J7', 300),
('S5', 'P2', 'J2', 200),
('S5', 'P2', 'J4', 100),
('S5', 'P5', 'J5', 500),
('S5', 'P5', 'J7', 100),
('S5', 'P6', 'J2', 200),
('S5', 'P1', 'J4', 100),
('S5', 'P3', 'J4', 200),
('S5', 'P4', 'J4', 800),
```

```
('S5', 'P5', 'J4', 400),
('S5', 'P6', 'J4', 500);

-- Selecting data from tables for checking
SELECT * FROM Suppliers;
SELECT * FROM Parts;
SELECT * FROM Jobs;
SELECT * FROM Shipments;
```

- **Using SQL Queries:**

Q1: List the part number for every part that is shipped by more than one supplier.

```
SELECT pnum
FROM Shipments
GROUP BY pnum
HAVING COUNT(DISTINCT snum) > 1;
```

Q2: Find the average weight of all parts.

```
SELECT AVG(weight) AS average_weight
FROM Parts;
```

Q3: For each part, list the part number and the total quantity in which that part is shipped and order the results in descending order of the total quantity shipped. Name the total quantity shipped in the result as total Shipped.

```
SELECT pnum, SUM(quantity) AS total_shipped
FROM Shipments
GROUP BY pnum
ORDER BY total_shipped DESC;
```

Q4: List only the names of those suppliers who ship a part that weighs more than 200.

```
SELECT DISTINCT sname
FROM Suppliers
JOIN Shipments ON Suppliers.snum = Shipments.snum
JOIN Parts ON Shipments.pnum = Parts.pnum
WHERE weight > 200;
```

Q5: List the names of those cities in which both a supplier and a job are located.

```
SELECT city
FROM Suppliers
INTERSECT
SELECT city
```

```
FROM Jobs;
```

Q6: List the names of those jobs that receive a shipment from supplier number S1.

```
SELECT DISTINCT jname
FROM Jobs
JOIN Shipments ON Jobs.jnum = Shipments.jnum
WHERE snum = 'S1';
```

Q7: List the names of those parts that are not shipped to any job.

```
SELECT pname
FROM Parts
WHERE pnum NOT IN (SELECT pnum FROM Shipments);
```

Q8: List the names of those suppliers who ship part number P2 to any job.

```
SELECT DISTINCT sname
FROM Suppliers
JOIN Shipments ON Suppliers.snum = Shipments.snum
WHERE pnum = 'P2';
```

Q9: List the names of those suppliers who ship part at least one red part to any job.

```
SELECT DISTINCT sname
FROM Suppliers
JOIN Shipments ON Suppliers.snum = Shipments.snum
JOIN Parts ON Shipments.pnum = Parts.pnum
WHERE color = 'RED';
```

Q10: List the part number for every part that is shipped more than once (the part must be shipped more than one time).

```
SELECT pnum
FROM Shipments
GROUP BY pnum
HAVING COUNT(pnum) > 1;
```

- ## Using Relational Algebra:

Q1: List the part number for every part that is shipped by more than one supplier.

$$\pi_{pnum}(\sigma_{(\Im\text{COUNT}((snum)>1))}(Shipments \bowtie_{Shipments.pnum=Parts.pnum} Parts))$$

Q2: Find the average weight of all parts.

$$\pi_{\text{weight}}\left(\Im(\text{AVG}(\text{weight})(\text{Parts}))\right)$$

Q3: For each part, list the part number and the total quantity in which that part is shipped and order the results in descending order of the total quantity shipped. Name the total quantity shipped in the result as total Shipped.

$$\pi_{pnum,\text{total\_shipped}}\left(\gamma_{pnum},\Im_{(SUM(quantity))\rightarrow\text{total\_shipped}}(\text{Shipments})\right)$$

Q4: List only the names of those suppliers who ship a part that weighs more than 200.

$$\pi_{\text{sname}}(\sigma_{\text{weight}>200}(\text{Suppliers}\bowtie_{\text{snum}}(\text{Shipments}\bowtie_{\text{pnum}}(\text{Parts}))))$$

Here, the Joins only have **snum** and **pnum** as attributes. This is a Natural Join which can also be represented as * because both tables Shipments and Parts have a column **pnum** and both tables Suppliers and Shipments have a column **snum**. Natural join can only be done if both Tables have columns with the same name and datatype. This is the reason there is no equality operator in the join (because it is not a theta join, rather a natural join).

Q5: List the names of those cities in which both a supplier and a job are located.

$$\pi_{\text{city}}(\text{Suppliers})\cap\pi_{\text{city}}(\text{Jobs})$$

Q6: List the names of those jobs that receive a shipment from supplier number S1.

$$\pi_{jname}\left(\sigma_{snum='S1'}(Jobs\bowtie_{Jobs.jnum=Shipments.jnum}Shipments)\right)$$

Q7: List the names of those parts that are not shipped to any job.

$$\pi_{pname}(Parts-(\pi_{pnum}(Shipments)))$$

Q8: List the names of those suppliers who ship part number P2 to any job.

$$\pi_{sname}\left(\sigma_{pnum='P2'}(Suppliers\bowtie_{Suppliers.snum=Shipments.snum}Shipments)\right)$$

Q9: List the names of those suppliers who ship part at least one red part to any job.

$$\pi_{sname}\left(\sigma_{color='RED'}(Suppliers\bowtie_{Suppliers.snum=Shipments.snum}Shipments\bowtie_{Shipments.pnum=Parts.pnum}Parts)\right)$$

Q10: List the part number for every part that is shipped more than once (the part must be shipped more than one time).

$$\pi_{pnum}\left(\gamma_{pnum}(\Im_{COUNT(pnum)>1})(Shipments)\right)$$

Note: The `SQL File` named `Supply_Chain_System.sql` is also submitted on Google Classroom, you can view all the Queries for Question 01 from there as well.

Question - 02:

Firstly create the Database and add data using these SQL Queries and adding data into tables:

```
use master
GO
CREATE DATABASE [Workforce_Management_System];
GO
USE [Workforce_Management_System];

-- Create Tables with their Schema
CREATE TABLE [Jobs]
(
    [JOB_ID] VARCHAR(100) PRIMARY KEY NOT NULL,
    JOB_TITLE VARCHAR(100) NOT NULL,
    MIN_SALARY INT NOT NULL,
    MAX_SALARY INT NOT NULL
);

CREATE TABLE [Locations]
(
    [LOCATION_ID] INT PRIMARY KEY NOT NULL,
    STREET_ADDRESS VARCHAR(100) NOT NULL,
    POSTAL_CODE VARCHAR(100) DEFAULT NULL,
    CITY VARCHAR(100) NOT NULL,
    STATE_PROVINCE VARCHAR(100) DEFAULT NULL,
    COUNTRY_ID CHAR(2) NOT NULL
);

CREATE TABLE [Departments]
(
    [DEPARTMENT_ID] INT PRIMARY KEY NOT NULL,
    DEPARTMENT_NAME VARCHAR(100) NOT NULL,
    MANAGER_ID INT NOT NULL,
    LOCATION_ID INT NOT NULL
    FOREIGN KEY (LOCATION_ID) REFERENCES Locations(LOCATION_ID)
);

CREATE TABLE [Employee]
(
    [EMPLOYEE_ID] INT PRIMARY KEY NOT NULL,
    FIRST_NAME VARCHAR(100) NOT NULL,
    LAST_NAME VARCHAR(100) NOT NULL,
    EMAIL VARCHAR(100) NOT NULL,
    PHONE_NUMBER VARCHAR(100) NOT NULL,
    HIRE_DATE DATE NOT NULL,
    JOB_ID VARCHAR(100) NOT NULL,
    SALARY INT NOT NULL,
```

```
    COMMISION_PCT INT NOT NULL DEFAULT 0,
    MANAGER_ID INT NOT NULL,
    DEPARTMENT_ID INT NOT NULL
    FOREIGN KEY (JOB_ID) REFERENCES Jobs(JOB_ID),
    FOREIGN KEY (DEPARTMENT_ID) REFERENCES Departments(DEPARTMENT_ID)
);

-- Entering Values into the Tables
INSERT INTO Jobs(JOB_ID, JOB_TITLE, MIN_SALARY, MAX_SALARY) VALUES
('AD_PRES', 'President', 20000, 40000),
('AD_VP', 'Administration Vice President', 15000, 30000),
('AD_ASST', 'Administration Assistant', 3000, 6000),
('FI_MGR', 'Finance Manager', 8200, 16000),
('FI_ACCOUNT', 'Accountant', 4200, 9000),
('AC_MGR', 'Accounting Manager', 8200, 16000),
('AC_ACCOUNT', 'Public Accountant', 4200, 9000),
('SA_MAN', 'Sales Manager', 10000, 20000),
('SA_REP', 'Sales Representative', 6000, 12000),
('PU_MAN', 'Purchasing Manager', 8000, 15000),
('PU_CLERK', 'Purchasing Clerk', 2500, 5500),
('ST_MAN', 'Stock Manager', 5500, 8500),
('ST_CLERK', 'Stock Clerk', 2000, 5000),
('SH_CLERK', 'Shipping Clerk', 2500, 5500),
('IT_PROG', 'Programmer', 4000, 10000),
('MK_MAN', 'Marketing Manager', 9000, 15000),
('MK_REP', 'Marketing Representative', 4000, 9000),
('HR_REP', 'Human Resources Representative', 4000, 9000),
('PR_REP', 'Public Relations Representative', 4500, 10500);

INSERT INTO Locations(LOCATION_ID, STREET_ADDRESS, POSTAL_CODE, CITY,
STATE_PROVINCE, COUNTRY_ID) VALUES
(1000, '1297 Via Cola di Rie', '00989', 'Roma', NULL, 'IT'),
(1100, '93091 Calle della Testa', '10934', 'Venice', NULL, 'IT'),
(1200, '2017 Shinjuku-ku', '1689', 'Tokyo', 'Tokyo Perfecture', 'JP'),
(1300, '9450 Kamiya-cho', '6823', 'Hiroshima', NULL, 'JP'),
(1400, '2014 Jabberwocky Rd', '26192', 'Southlake', 'Texas', 'US'),
(1500, '2011 Interiors Blvd', '99236', 'South San Francisco', 'California',
'US'),
(1600, '2007 Zagora St', '50090', 'South Brunswick', 'New Jersey', 'US'),
(1700, '2004 Charade Rd', '98199', 'Seattle', 'Washington', 'US'),
(1800, '147 Spadina Ave', 'M5V 2L7', 'Toronto', 'Ontario', 'CA'),
(1900, '6092 Boxwood St', 'YSW 9T2', 'Whitehorse', 'Yukon', 'CA'),
(2000, '40-5-12 Laogianggen', '190518', 'Beijing', NULL, 'CN'),
(2100, '1298 Vileparle (E)', '490231', 'Mumbai', 'Maharashtra', 'IN'),
(2200, '12-98 Victoria Street', '2901', 'Sydney', 'New South Wales', 'AU'),
(2300, '198 Clementi North', '540198', 'Singapore', NULL, 'SG'),
(2400, '8204 Arthur St', NULL, 'London', NULL, 'UK'),
(2500, 'Magdalen Centre, The Oxford Science Park', 'OX9 9ZB', 'Oxford',
```

```
'Oxford', 'UK'),
(2600, '9702 Chester Road', '9629850293', 'Stretford', 'Manchester', 'UK'),
(2700, 'Schwanthalerstr. 7031', '80925', 'Munich', 'Bavaria', 'DE'),
(2800, 'Rua Frei Caneca 1360', '01307-002', 'Sao Paulo', 'Sao Paulo', 'BR'),
(2900, '20 Rue des Corps-Saints', '1730', 'Geneva', 'Geneve', 'CH'),
(3000, 'Murtenstrasse 921', '3095', 'Bern', 'BE', 'CH'),
(3100, 'Pieter Breughelstraat 837', '3029SK', 'Utrecht', 'Utrecht', 'NL'),
(3200, 'Mariano Escobedo 9991', '11932', 'Mexico City', 'Distrito Federal
Mexico', 'MX');

INSERT INTO Departments(DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID,
LOCATION_ID) VALUES
(10, 'Administration', 200, 1700),
(20, 'Marketing', 201, 1800),
(30, 'Purchasing', 114, 1700),
(40, 'Human Resources', 203, 2400),
(50, 'Shipping', 121, 1500),
(60, 'IT', 103, 1400),
(70, 'Public Relations', 204, 2700),
(80, 'Sales', 145, 2500),
(90, 'Executive', 100, 1700),
(100, 'Finance', 108, 1700),
(110, 'Accounting', 205, 1700),
(120, 'Treasury', 0, 1700),
(130, 'Corporate Tax', 0, 1700),
(140, 'Control And Credit', 0, 1700),
(150, 'Shareholder Services', 0, 1700),
(160, 'Benefits', 0, 1700);

INSERT INTO Employee(EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISION_PCT, MANAGER_ID,
DEPARTMENT_ID) VALUES
(100, 'Steven', 'King', 'SKING', '515.123.4567', '1987-06-17', 'AD_PRES',
24000, 0, 0, 90),
(101, 'Neena', 'Kochhar', 'NKOCHHAR', '515.123.4568', '1987-06-18', 'AD_VP',
17000, 0, 100, 90),
(102, 'Lex', 'De Haan', 'LDEHAAN', '515.123.4569', '1987-06-19', 'AD_VP',
17000, 0, 100, 90),
(103, 'Alexander', 'Hunold', 'AHUNOLD', '590.423.4567', '1987-06-20',
'IT_PROG', 9000, 0, 102, 60),
(104, 'Bruce', 'Ernst', 'BERNST', '590.423.4568', '1987-06-21', 'IT_PROG',
6000, 0, 103, 60),
(105, 'David', 'Austin', 'DAUSTIN', '590.423.4569', '1987-06-22', 'IT_PROG',
4800, 0, 103, 60),
(106, 'Valli', 'Pataballa', 'VPATABAL', '590.423.4560', '1987-06-23',
'IT_PROG', 4800, 0, 103, 60),
(107, 'Diana', 'Lorentz', 'DLORENTZ', '590.423.5567', '1987-06-24',
'IT_PROG', 4200, 0, 103, 60),
```

```
(108, 'Nancy', 'Greenberg', 'NGREENBE', '515.124.4569', '1987-06-25',
'FI_MGR', 12000, 0, 101, 100),
(109, 'Daniel', 'Faviet', 'DFAVIET', '515.124.4169', '1987-06-26',
'FI_ACCOUNT', 9000, 0, 108, 100),
(110, 'John', 'Chen', 'JCHEN', '515.124.4269', '1987-06-27', 'FI_ACCOUNT',
8200, 0, 108, 100),
(111, 'Ismael', 'Sciarra', 'ISCIARRA', '515.124.4369', '1987-06-28',
'FI_ACCOUNT', 7700, 0, 108, 100),
(112, 'Jose Manuel', 'Urman', 'JMURMAN', '515.124.4469', '1987-06-29',
'FI_ACCOUNT', 7800, 0, 108, 100),
(113, 'Luis', 'Popp', 'LPOPP', '515.124.4567', '1987-06-30', 'FI_ACCOUNT',
6900, 0, 108, 100),
(114, 'Den', 'Raphaely', 'DRAPHEAL', '515.127.4561', '1987-07-01', 'PU_MAN',
11000, 0, 100, 30),
(115, 'Alexander', 'Khoo', 'AKHOO', '515.127.4562', '1987-07-02',
'PU_CLERK', 3100, 0, 114, 30),
(116, 'Shelli', 'Baida', 'SBAIDA', '515.127.4563', '1987-07-03', 'PU_CLERK',
2900, 0, 114, 30),
(117, 'Sigal', 'Tobias', 'STOBIAS', '515.127.4564', '1987-07-04',
'PU_CLERK', 2800, 0, 114, 30),
(118, 'Guy', 'Himuro', 'GHIMURO', '515.127.4565', '1987-07-05', 'PU_CLERK',
2600, 0, 114, 30),
(119, 'Karen', 'Colmenares', 'KCOLMENA', '515.127.4566', '1987-07-06',
'PU_CLERK', 2500, 0, 114, 30),
(120, 'Matthew', 'Weiss', 'MWEISS', '650.123.1234', '1987-07-07', 'ST_MAN',
8000, 0, 100, 50),
(121, 'Adam', 'Fripp', 'AFRIPP', '650.123.2234', '1987-07-08', 'ST_MAN',
8200, 0, 100, 50),
(122, 'Payam', 'Kaufling', 'PKAUFLIN', '650.123.3234', '1987-07-09',
'ST_MAN', 7900, 0, 100, 50),
(123, 'Shanta', 'Vollman', 'SVOLLMAN', '650.123.4234', '1987-07-10',
'ST_MAN', 6500, 0, 100, 50),
(124, 'Kevin', 'Mourgos', 'KMOURGOS', '650.123.5234', '1987-07-11',
'ST_MAN', 5800, 0, 100, 50),
(125, 'Julia', 'Nayer', 'JNAYER', '650.124.1214', '1987-07-12', 'ST_CLERK',
3200, 0, 120, 50);
```

- **SQL Queries:**

Q1: find the name (first_name, last_name) and the salary of the employees who have a higher salary than the employee whose last_name = 'Bull'.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM Employee
WHERE SALARY > (SELECT SALARY FROM Employee WHERE LAST_NAME = 'Bull');
```

Q2: find the name (first_name, last_name) of all employees who work in the IT department.

```
SELECT FIRST_NAME, LAST_NAME
FROM Employee
WHERE DEPARTMENT_ID = (SELECT DEPARTMENT_ID FROM Departments WHERE
DEPARTMENT_NAME = 'IT');
```

Q3: find the name (first_name, last_name) of the employees who have a manager and worked in a USA based department.

```
SELECT DISTINCT FIRST_NAME, LAST_NAME
FROM Employee
WHERE MANAGER_ID IS NOT NULL AND DEPARTMENT_ID IN
(SELECT DEPARTMENT_ID FROM Departments WHERE LOCATION_ID IN
(SELECT LOCATION_ID FROM Locations WHERE COUNTRY_ID = 'US'));
```

Q4: find those employees who earn more than the average salary. Return employee ID, first name, last name.

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME
FROM Employee
WHERE SALARY > (SELECT AVG(SALARY) FROM Employee);
```

Q5: find those employees whose department is located at 'Toronto'. Return first name, last name, employee ID, job ID.

```
SELECT e.FIRST_NAME, e.LAST_NAME, e.EMPLOYEE_ID, e.JOB_ID
FROM Employee e
JOIN Departments d ON e.DEPARTMENT_ID = d.DEPARTMENT_ID
JOIN Locations l ON d.LOCATION_ID = l.LOCATION_ID
WHERE l.CITY = 'Toronto';
```

Q6: find those employees who report to that manager whose first name is 'Payam'. Return first name, last name, employee ID and salary.

```
SELECT FIRST_NAME, LAST_NAME, EMPLOYEE_ID, SALARY
FROM Employee
WHERE MANAGER_ID IN
(SELECT EMPLOYEE_ID FROM Employee WHERE FIRST_NAME = 'Payam');
```

Q7: find all those departments where at least one employee is employed. Return department name.

```
SELECT DEPARTMENT_NAME
FROM Departments
```

```
WHERE DEPARTMENT_ID IN
(SELECT DEPARTMENT_ID FROM Employee);
```

Q8: find those employees who do not work in the departments where managers' IDs are between 100 and 200 (Begin and end values are included.). Return all the fields of the employees.

```
SELECT *
FROM Employee
WHERE DEPARTMENT_ID NOT IN (
    SELECT DEPARTMENT_ID
    FROM Departments
    WHERE MANAGER_ID BETWEEN 100 AND 200
);
```

Q9: From the following table, find those employees whose salary matches the lowest salary of any of the departments. Return first name, last name and department ID.

```
SELECT FIRST_NAME, LAST_NAME, DEPARTMENT_ID
FROM Employee
WHERE SALARY = (SELECT MIN(SALARY) FROM Employee);
```

Q10: find the name (first_name, last_name) of the employees who are managers.

```
SELECT FIRST_NAME, LAST_NAME
FROM Employee
WHERE EMPLOYEE_ID IN (SELECT MANAGER_ID FROM Departments);
```

Q11: find those employees whose salary is lower than that of employees whose job title is 'MK_MAN'. Exclude employees of Job title 'MK_MAN'. Return employee ID, first name, last name, job ID.

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, JOB_ID
FROM Employee
WHERE SALARY < (
    SELECT SALARY
    FROM Employee
    WHERE JOB_ID = 'MK_MAN'
)
AND JOB_ID != 'MK_MAN';
```

Q12: Find the name (first_name, last_name), and salary of the employees whose salary is greater than the average salary.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM Employee
```

```
WHERE SALARY > (
    SELECT AVG(SALARY)
    FROM Employee
);
```

Q13: Find the name (first_name, last_name), and salary of the employees whose salary is equal to the minimum salary for their job grade.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM Employee
WHERE SALARY = (
    SELECT MIN_SALARY
    FROM Jobs
    WHERE JOB_ID = Employee.JOB_ID
);
```

Q14: Find the name (first_name, last_name), and salary of the employees who earns more than the average salary and works in any of the IT departments.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM Employee
WHERE SALARY > (
    SELECT AVG(SALARY)
    FROM Employee
)
AND DEPARTMENT_ID IN (
    SELECT DEPARTMENT_ID
    FROM Departments
    WHERE DEPARTMENT_NAME = 'IT'
);
```

Q15: Find the name (first_name, last_name), and salary of the employees who earns more than the earning of Mr. Bell.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM Employee
WHERE SALARY > (
    SELECT SALARY
    FROM Employee
    WHERE LAST_NAME = 'Bell'
);
```

Q16: .Find the name (first_name, last_name), and salary of the employees who earn the same salary as the minimum salary for all departments.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
```

```
FROM Employee
WHERE SALARY = (
    SELECT MIN(SALARY)
    FROM Employee
);
```

Q17: Find the name (first_name, last_name), and salary of the employees whose salary is greater than the average salary of all departments.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM Employee
WHERE SALARY > (
    SELECT AVG(SALARY)
    FROM Employee
);
```

Q18: Find the 3rd maximum salary in the employees table.

```
SELECT DISTINCT SALARY
FROM Employee
ORDER BY SALARY DESC
OFFSET 2 ROWS
FETCH NEXT 1 ROWS ONLY;
```

- ## Using Relational Algebra:

Q1: find the name (first_name, last_name) and the salary of the employees who have a higher salary than the employee whose last_name = 'Bull'.

$$\pi_{\text{FIRST\_NAME, LAST\_NAME, SALARY}} \left(\sigma_{\text{SALARY}>(\pi_{\text{SALARY}}(\sigma_{\text{LAST\_NAME = 'Bull'}}(\text{Employee})))}(\text{Employee})\right)$$

Q2: find the name (first_name, last_name) of all employees who work in the IT department.

$$\pi_{\text{FIRST\_NAME, LAST\_NAME}} \left(\sigma_{\text{DEPARTMENT\_ID}=(\pi_{\text{DEPARTMENT\_ID}}(\sigma_{\text{DEPARTMENT\_NAME = 'IT'}}(\text{Departments})))}(\text{Employee})\right)$$

Q3: find the name (first_name, last_name) of the employees who have a manager and worked in a USA based department.

$$\pi_{first\_name,last\_name}(Employee \bowtie_{Employee.manager\_id=manager.employee\_id \cap location.country\_id='US'} (\sigma_{country\_id='US'}(Locations) \bowtie Departments))$$

Here the Final Join Symbol is a Natural Join, so it has no equality operators.

Q4: find those employees who earn more than the average salary. Return employee ID, first name, last name.

$$\pi_{\text{EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME}}\left(\sigma_{\text{SALARY}>\left(\Im_{\text{AVG(SALARY)}}(\text{Employee})\right)}(\text{Employee})\right)$$

Q5: find those employees whose department is located at 'Toronto'. Return first name, last name, employee ID, job ID.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,EMPLOYEE\_ID,JOB\_ID}}\left(\sigma_{\text{CITY}='Toronto'}\left(\text{Employee} \bowtie (\text{Departments} \bowtie \text{Locations})\right)\right)$$

Here, all three Joins are Natural Joins which can also be represented by * Symbol as given:

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,EMPLOYEE\_ID,JOB\_ID}}\left(\sigma_{\text{CITY}='Toronto'}\left(\text{Employee} * (\text{Departments} * \text{Locations})\right)\right)$$

Q6: find those employees who report to that manager whose first name is 'Payam'. Return first name, last name, employee ID and salary.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,EMPLOYEE\_ID,SALARY}}\left(\sigma_{\text{MANAGER\_ID}\in\left(\pi_{\text{EMPLOYEE\_ID}}\left(\sigma_{\text{FIRST\_NAME}='Payam'}(\text{Employee})\right)\right)}(\text{Employee})\right)$$

Q7: find all those departments where at least one employee is employed. Return department name.

$$\pi_{\text{DEPARTMENT\_NAME}}\left(\sigma_{\text{DEPARTMENT\_ID}\in(\pi_{\text{DEPARTMENT\_ID}}(\text{Employee}))}(\text{Departments})\right)$$

Please note that this can be done by Union and Intersection as well. But that will take a lot of lines, so I used the ***belongs to*** symbol for it.

Q8: find those employees who do not work in the departments where managers' IDs are between 100 and 200 (Begin and end values are included.). Return all the fields of the employees.

$$\text{Temp} \leftarrow \pi_{\text{DEPARTMENT\_ID}}(\text{Departments}) - \pi_{\text{DEPARTMENT\_ID}}(\text{Employee})$$

$$\pi_{\text{EMPLOYEE\_ID,FIRST\_NAME,LAST\_NAME,EMAIL,PHONE\_NUMBER,HIRE\_DATE,}}$$
$$\text{JOB\_ID,SALARY,COMMISION\_PCT,MANAGER\_ID,DEPARTMENT\_ID}$$
$$\left(\sigma_{\text{DEPARTMENT\_ID}\in\text{Temp}}(\text{Employee})\right)$$

The first equation is complete, the 2nd equation is cut into the last 3 lines because of lack of space.

It can also be written as:

$$\text{Employee} - \text{DEPARTMENT\_ID NOT IN}\left(\pi_{\text{DEPARTMENT\_ID}}\left(\sigma_{\text{MANAGER\_ID}\geq 100 \wedge \text{MANAGER\_ID}\leq 200}(\text{Departments})\right)\right)$$

Q9: From the following table, find those employees whose salary matches the lowest salary of any of the departments. Return first name, last name and department ID.

$$\Pi_{\text{FIRST\_NAME,LAST\_NAME,DEPARTMENT\_ID}}\left(\sigma_{\text{SALARY}=\left(\Im_{\text{MIN (SALARY)}}(\text{Employee})\right)}\right)$$

There is an Equal Operator after the Select **SALARY** statement. Because of the lines being

thin, it looks like Set Difference (-).

Q10: find the name (first_name, last_name) of the employees who are managers.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME}}\left(\sigma_{\text{EMPLOYEE\_ID}\in\left(\pi_{\text{MANAGER\_ID}}(\text{Departments})\right)}(\text{Employee})\right)$$

Q11: find those employees whose salary is lower than that of employees whose job title is 'MK_MAN'. Exclude employees of Job title 'MK_MAN'. Return employee ID, first name, last name, job ID.

$$\pi_{\text{EMPLOYEE\_ID,FIRST\_NAME,LAST\_NAME,JOB\_ID}}\left(\sigma_{\text{SALARY}<}\right.$$

$$\left.\left(\pi_{\text{SALARY}}(\sigma_{\text{JOB\_ID}='MK\_MAN'}(\text{Employee}))\right)\wedge\text{JOB\_ID}\neq'MK\_MAN'(\text{Employee})\right)$$

This is the same equation divided into 2 parts.

Q12: Find the name (first_name, last_name), and salary of the employees whose salary is greater than the average salary.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,SALARY}}\left(\sigma_{\text{SALARY}>(\Im_{AVERAGE(SALARY)}(\text{Employee}))}\right)$$

Q13: Find the name (first_name, last_name), and salary of the employees whose salary is equal to the minimum salary for their job grade.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,SALARY}}\left(\sigma_{\text{SALARY}=(\Im_{\text{MIN(SALARY))}}}(\text{Employee})\right)$$

Q14: Find the name (first_name, last_name), and salary of the employees who earns more than the average salary and works in any of the IT departments.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,SALARY}}\left(\sigma_{\text{SALARY}>(\Im_{\text{AVERAGE(SALARY)}})\wedge\text{DEPARTMENT\_ID}\in\left(\pi_{\text{DEPARTMENT\_ID}}\right.\right.$$

$$\left.\left.\left(\sigma_{\text{DEPARTMENT\_NAME}='IT'}(\text{Departments})\right)\right)(\text{Employee})\right)$$

This is a Single Equation divided into multiple lines.

Q15: Find the name (first_name, last_name), and salary of the employees who earns more than the earning of Mr. Bell.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,SALARY}}\left(\sigma_{\text{SALARY}>\pi_{\text{SALARY}}\left(\sigma_{\text{LAST\_NAME}='Bell'}(\text{Employee})\right)}(\text{Employee})\right)$$

Q16: .Find the name (first_name, last_name), and salary of the employees who earn the same salary as the minimum salary for all departments.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,SALARY}} \left(\sigma_{\text{SALARY}=(\Im_{\text{Min(Salary)}})} (\text{Employee}) \right)$$

Q17: Find the name (first_name, last_name), and salary of the employees whose salary is greater than the average salary of all departments.

$$\pi_{\text{FIRST\_NAME,LAST\_NAME,SALARY}} \left(\sigma_{\text{SALARY}>(\Im_{\text{AVERAGE(SALARY)}}(\text{Employee}))} (\text{Employee}) \right)$$

Q18: Find the 3rd maximum salary in the employees table.

$$\text{ThirdMaxSalary} \leftarrow \pi_{\text{SALARY}} (\text{Employee}) - \left(\Im_{\text{MAX}(\pi_{\text{SALARY}}(\text{Employee}))} \right)$$

$$\text{NewThirdMaxSalary} \leftarrow \text{ThirdMaxSalary} - \left(\Im_{\text{MAX(ThirdMaxSalary)}} \right)$$

$$\text{FinalThirdMaxSalary} \leftarrow \Im_{\text{MAX(NewThirdMaxSalary)}}$$

Note: The Relational Algebra equations written in more than 1 different lines are the same query. But, it was a long query. So, I divided it into multiple lines. Moreover, please check the `Workforce_Management_System.sql` file which is submitted on Google Classroom and contains all SQL Queries for Question 02.
