403-RDBMS-LAB MANUAL)

M.Com (CA) Semester-IV

Student database – Library database – Salary database – students enrollment – Book Dealer database – Bank database – Order Processing database.

M.C.Mouli, Department of Commerce & Computer

403-LAB: RELATIONAL DATABASEMANAGEMENTSYSTEMS
(For M.Com-Computer Applications- under CBCS)

Lab: 2 PPW: One Period Lab means 2 hours of Lab Session with SQL.

*****

1. The STUDENTDET AIL databases have a table with the following attributes.

STUDENT (regno: int, name: string, dob: date, marks: int)

i) Create the above table.
ii) Remove the existing attributes from the table.
iii) Change the data type of regno from integer to string.
iv) Add a new attribute phone number to the existing table.
v) Enter five tuples into the table.
vi) Display all the tuples of student table.

Solution:

i) Create the table:

```
CREATE TABLE STUDENT ( regno INT, name VARCHAR(255), dob DATE, marks INT );
```

ii) Remove existing attributes:

```
ALTER TABLE STUDENT DROP COLUMN regno, DROP COLUMN name, DROP COLUMN dob, DROP
COLUMN marks;
```

iii) Change the data type of regno from integer to string:

```
ALTER TABLE STUDENT ADD regno VARCHAR(20);
```

iv) Add a new attribute phone number:

```
ALTER TABLE STUDENT ADD phone_number VARCHAR(20);
```

v) Enter five tuples into the table:

```
INSERT INTO STUDENT (regno, name, dob, marks, phone_number) VALUES
('001', 'John', '2000-01-01', 85, '123-456-7890'),
('002', 'Emma', '2001-03-15', 78, '987-654-3210'),
('003', 'Michael', '2002-05-20', 90, '555-555-5555'),
('004', 'Sophia', '2003-07-10', 92, '999-888-7777'),
('005', 'William', '2004-09-30', 87, '444-333-2222');
```

vi) Display all tuples of the student table:

```
SELECT * FROM STUDENT;
```

2. A LIBRARY database has a table with the following attributes.

LIBRARY (bookid:int, title:string, author:string, publication:string, yearpub:int, price:real)

i) Create the above table. Enter five tuples into the table
ii) Display all the tuples in student table.
iii) Display the different publishers from the list.
iv) Arrange the tuples in the alphabetical order of the book titles.
v) List the details of all the books whose price ranges between Rs. 100 and Rs. 300

Solution:

i) Create the table and enter five tuples:

```
CREATE TABLE LIBRARY ( bookid INT, title VARCHAR(255),
author VARCHAR(255), publication VARCHAR(255), yearpub INT, price REAL );

INSERT INTO LIBRARY (bookid, title, author, publication, yearpub, price) VALUES
```

(1, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Scribner', 1925, 250.00),
(2, 'To Kill a Mockingbird', 'Harper Lee', 'J.B. Lippincott & Co.', 1960, 200.00),
(3, '1984', 'George Orwell', 'Secker & Warburg', 1949, 150.00),
(4, 'The Catcher in the Rye', 'J.D. Salinger', 'Little, Brown and Company', 1951, 180.00),
(5, 'Pride and Prejudice', 'Jane Austen', 'T. Egerton, Whitehall', 1813, 300.00);

ii) Display all tuples in the library table:
SELECT * FROM LIBRARY;

iii) Display different publishers from the list:
SELECT DISTINCT publication FROM LIBRARY;

iv) Arrange the tuples in alphabetical order of the book titles:
SELECT * FROM LIBRARY ORDER BY title;

v) List the details of all the books whose price ranges between Rs. 100 and Rs. 300:
SELECT * FROM LIBRARY WHERE price BETWEEN 100 AND 300;
These queries should accomplish the tasks you outlined for the LIBRARY database.

3. The SALARY database of an organization has a table with the following attributes.
EMPSALARY (empcode: int, empname: string, dob: date, department: string, salary:real)
i) Create the above table. Enter five tuples into the table.
ii) Display all the number of employees working in each department.
iii) Find the sum of the salaries of all employees.
iv) Find the sum and average of the salaries of employees of a particular department.
v) Find the least and highest salaries that an employee draws.

Solution:
i) Create the table and enter five tuples:
CREATE TABLE EMPSALARY (empcode INT, empname VARCHAR(255), dob DATE,
department VARCHAR(255), salary REAL);

INSERT INTO EMPSALARY (empcode, empname, dob, department, salary) VALUES
(1, 'John Doe', '1990-05-15', 'Sales', 50000.00),
(2, 'Jane Smith', '1985-09-20', 'Marketing', 60000.00),
(3, 'Michael Johnson', '1988-07-10', 'HR', 55000.00),
(4, 'Emily Brown', '1992-03-25', 'Finance', 65000.00),
(5, 'David Wilson', '1995-01-30', 'IT', 70000.00);

ii) Display the number of employees working in each department:
SELECT department, COUNT(*) AS num_employees FROM EMPSALARY GROUP BY department;

iii) Find the sum of the salaries of all employees:
SELECT SUM (salary) AS total_salary FROM EMPSALARY;
iv) Find the sum and average of the salaries of employees of a particular department (let's say 'Sales'):
SELECT SUM (salary) AS total_salary_sales, AVG(salary) AS average_salary_sales FROM EMPSALARY
WHERE department = 'Sales';
v) Find the least and highest salaries that an employee draws:
SELECT MIN (salary) AS min_salary, MAX(salary) AS max_salary FROM EMPSALARY;
These queries should accomplish the tasks you outlined for the SALARY database.

4. Consider the insurance database given below.
PERSON(driver-id-no: string, name: string, address: string)
CAR(regno: string, model: string, year: int)
ACCIDENT(report-no: int, date: date, location: String)
OWNS(driver-id-no: string, regno: string)
PARTICIPATED (driver-id-no: string, regno: string, report-no: int, damage-amount: int)
i) Create the above tables by properly specifying the primary keys and the foreign keys.
Enter at least five tuples or each relation.
ii) Demonstrate how you:
a) Update the damage amount for the car with a specific regno in the accident with Report no 12 to 25000.
b) Add a new accident to the database.
iii) Find total number of people who owned cars that were involved in accidents in 2012
iv) Find the number of accidents in which cars belonging to a specific model were involved.

Solution: Here's how you can create the tables and perform the operations:
i) Create tables with primary keys and foreign keys:

```
CREATE TABLE PERSON ( driver_id_no VARCHAR(20) PRIMARY KEY, name VARCHAR(255), address
VARCHAR(255) );

CREATE TABLE CAR ( regno VARCHAR(20) PRIMARY KEY, model VARCHAR(255), year INT );

CREATE TABLE ACCIDENT ( report_no INT PRIMARY KEY, date DATE, location VARCHAR(255) );

CREATE TABLE OWNS ( driver_id_no VARCHAR(20), regno VARCHAR(20), PRIMARY KEY
(driver_id_no, regno), FOREIGN KEY (driver_id_no) REFERENCES PERSON(driver_id_no),
FOREIGN KEY (regno) REFERENCES CAR(regno) );

CREATE TABLE PARTICIPATED ( driver_id_no VARCHAR(20), regno VARCHAR(20),
report_no INT, damage_amount INT,
PRIMARY KEY (driver_id_no, regno, report_no),
FOREIGN KEY (driver_id_no) REFERENCES PERSON(driver_id_no),
FOREIGN KEY (regno) REFERENCES CAR(regno),
FOREIGN KEY (report_no) REFERENCES ACCIDENT(report_no) );
```

Inserting tuples into PERSON table

```
INSERT INTO PERSON (driver_id_no, name, address) VALUES
('D001', 'John', '123 Main St'),
('D002', 'Emma', '456 Elm St'),
('D003', 'Michael', '789 Oak St'),
('D004', 'Sophia', '101 Pine St'),
('D005', 'William', '202 Maple St');
```

Inserting tuples into CAR table

```
INSERT INTO CAR (regno, model, year) VALUES
('ABC123', 'Toyota', 2010),
('DEF456', 'Honda', 2012),
('GHI789', 'Ford', 2015),
('JKL012', 'Chevrolet', 2018),
('MNO345', 'Toyota', 2012);
```

Inserting tuples into ACCIDENT table

```
INSERT INTO ACCIDENT (report_no, date, location) VALUES
(10, '2012-05-15', 'Intersection A'),
 (11, '2012-08-20', 'Highway B'),
 (12, '2012-10-10', 'Street C'),
(13, '2013-02-28', 'Road D'),
(14, '2013-06-05', 'Avenue E');
```

Inserting tuples into OWNS table

```
INSERT INTO OWNS (driver_id_no, regno) VALUES
('D001', 'ABC123'), ('D002', 'DEF456'), ('D003', 'GHI789'),
('D004', 'JKL012'), ('D005', 'MNO345');
```

Inserting tuples into PARTICIPATED table

```
INSERT INTO PARTICIPATED (driver_id_no, regno, report_no, damage_amount) VALUES
('D001', 'ABC123', 10, 5000), ('D002', 'DEF456', 11, 8000), ('D003', 'GHI789', 12, 12000),
('D004', 'JKL012', 13, 10000), ('D005', 'MNO345', 14, 6000);
```

ii) Demonstrate updating damage amount and adding a new accident:
a) Update the damage amount for the car with a specific regno in the accident with Report no 12 to 25000:

```
UPDATE PARTICIPATED SET damage_amount = 25000 WHERE regno = 'GHI789' AND report_no =
12;
```

b) Add a new accident to the database:

```
INSERT INTO ACCIDENT (report_no, date, location) VALUES (15, '2014-04-10', 'Street F');
```

iii) Find the total number of people who owned cars that were involved in accidents in 2012:

```
SELECT COUNT(DISTINCT driver_id_no) AS num_people FROM OWNS
JOIN
PARTICIPATED ON OWNS.regno = PARTICIPATED.regno JOIN ACCIDENT ON
PARTICIPATED.report_no = ACCIDENT.report_no
WHERE YEAR(ACCIDENT.date) = 2012;
```

iv) Find the number of accidents in which cars belonging to a specific model were involved (e.g., 'Toyota'):

```
SELECT COUNT(*) AS num_accidents FROM PARTICIPATED JOIN CAR ON
PARTICIPATED.regno = CAR.regno
WHERE CAR.model = 'Toyota';
```

These queries should cover all the operations you mentioned for the insurance database.

5. Consider the following database of student’s enrollment in courses and books adopted for each course.

 STUDENT(regno: string, name: string, major: string, bdate: date)
 COURSE(course-no: int cname: string, dept: string)
 ENROLL(reg-no: string, course-no: int, sem: int, marks: int)
 BOOK-ADOPTION (course-no: int, sem: int, book-isbn: int)
 TEXT(book-isbn: int, book-title: string, publisher: string, author: string.

i) Create the above tables by properly specifying the primary keys and the foreign keys, Enter at least five tuples for each relation.
ii) Demonstrate how you add a new text book to the database and make this book be adopted by some department.
iii) Produce a list of text books (include Course-no, book-isbn, book-title) in the alphabetical order for Courses offered by the 'Computer Science' department that use more than two books.
iv) List any department that has all its adopted books published by a specific publisher.

Solution:
Here's how you can create the tables, enter tuples, and perform the operations:
i) Create tables with primary keys and foreign keys, and enter at least five tuples for each relation:

```
CREATE TABLE STUDENT ( regno VARCHAR(20) PRIMARY KEY,
name VARCHAR(255), major VARCHAR(255), bdate DATE );

CREATE TABLE COURSE ( course_no INT PRIMARY KEY,
cname VARCHAR(255), dept VARCHAR(255) );

CREATE TABLE ENROLL ( reg_no VARCHAR(20), course_no INT, sem INT, marks INT,
PRIMARY KEY (reg_no, course_no, sem),
FOREIGN KEY (reg_no) REFERENCES STUDENT(regno),
FOREIGN KEY (course_no) REFERENCES COURSE(course_no) );

CREATE TABLE BOOK_ADOPTION ( course_no INT, sem INT, book_isbn INT,
PRIMARY KEY (course_no, sem, book_isbn),
FOREIGN KEY (course_no) REFERENCES COURSE(course_no),
FOREIGN KEY (book_isbn) REFERENCES TEXT(book_isbn) );

CREATE TABLE TEXT ( book_isbn INT PRIMARY KEY, book_title VARCHAR(255), publisher
VARCHAR(255), author VARCHAR(255) );

Inserting tuples into STUDENT table
INSERT INTO STUDENT (regno, name, major, bdate) VALUES
('S001', 'John', 'Computer Science', '2000-01-01'),  ('S002', 'Emma', 'Engineering', '1999-05-15'),
('S003', 'Michael', 'Computer Science', '2001-03-20'),
('S004', 'Sophia', 'Mathematics', '1998-07-10'), ('S005', 'William', 'Computer Science', '2002-09-30');

Inserting tuples into COURSE table
INSERT INTO COURSE (course_no, cname, dept) VALUES
(101, 'Introduction to Computer Science', 'Computer Science'),
(102, 'Data Structures', 'Computer Science'), (201, 'Engineering Mathematics', 'Engineering'),
(202, 'Mechanics', 'Engineering'),(301, 'Linear Algebra', 'Mathematics');

Inserting tuples into ENROLL table
INSERT INTO ENROLL (reg_no, course_no, sem, marks) VALUES
('S001', 101, 1, 85), ('S002', 102, 1, 78), ('S003', 101, 1, 90),
('S004', 201, 1, 92), ('S005', 101, 1, 87);

Inserting tuples into TEXT table
INSERT INTO TEXT (book_isbn, book_title, publisher, author) VALUES
(1, 'Introduction to Algorithms', 'MIT Press', 'Thomas H. Cormen'),
```

(2, 'Data Structures and Algorithm Analysis in C', 'Addison-Wesley', 'Mark Allen Weiss'),
(3, 'Engineering Mathematics', 'Pearson', 'K.A. Stroud'),
(4, 'Statics and Mechanics of Materials', 'John Wiley & Sons', 'Russell C. Hibbeler'),
(5, 'Linear Algebra and Its Applications', 'Pearson', 'David C. Lay');

Inserting tuples into BOOK_ADOPTION table
INSERT INTO BOOK_ADOPTION (course_no, sem, book_isbn) VALUES
(101, 1, 1), (101, 1, 2), (102, 1, 2), (201, 1, 3), (301, 1, 5);

ii) Demonstrate how to add a new textbook to the database and make this book be adopted by some department:
Adding a new textbook
INSERT INTO TEXT (book_isbn, book_title, publisher, author) VALUES
(6, 'Operating System Concepts', 'John Wiley & Sons', 'Abraham Silberschatz');

Making the new book be adopted by a department (e.g., Computer Science)
INSERT INTO BOOK_ADOPTION (course_no, sem, book_isbn) VALUES (101, 2, 6);

iii) Produce a list of textbooks (including Course-no, book-isbn, book-title) in alphabetical order for Courses offered by the 'Computer Science' department that use more than two books:
SELECT BA.course_no, BA.book_isbn, T.book_title FROM BOOK_ADOPTION BA
JOIN TEXT T ON BA.book_isbn = T.book_isbn
JOIN COURSE C ON BA.course_no = C.course_no WHERE C.dept = 'Computer Science'
GROUP BY BA.course_no HAVING COUNT(*) > 2 ORDER BY T.book_title;

iv) List any department that has all its adopted books published by a specific publisher (e.g., 'Pearson'):

SELECT C.dept FROM COURSE C LEFT JOIN BOOK_ADOPTION BA
ON C.course_no = BA.course_no LEFT JOIN TEXT T ON BA.book_isbn = T.book_isbn
GROUP BY C.dept HAVING COUNT (*) = SUM (CASE WHEN T.publisher = 'Pearson'
THEN 1 ELSE 0 END);

These queries should cover all the operations you mentioned for the student enrollment and book adoption database.

6. The following tables are maintained by a book dealer
AUTHOR (author-id: int, name: string, city: string, country: string)
PUBLISHER (publisher-id: int name: string, city: string, country: string)
CATLOG (book-id: int, title : string, author-id: int, publisher-id: int, category: int, year: int, price: int)
CATEGORY (category-id: int, description: string)
ORDER-DETAILS (order-no: int, book-id: int, quantity: int)

i) Create above tables by properly specifying the primary keys and the foreign keys. Enter at least five tuples for each relation.
ii) Give the details of the authors who have 2 or more books in the catalog and the price of the books is greater than the average price of the books in the catalog and the year of publication is after 2010.
iii) Find the author of the book which has maximum sales.
iv) Demonstrate how to increase price of books published by specific publisher by 10%

Solution:

i) Creating tables with primary keys and foreign keys:

```
CREATE TABLE AUTHOR ( author_id INT PRIMARY KEY,
name VARCHAR(100), city VARCHAR(100),
country VARCHAR(100) );

CREATE TABLE PUBLISHER ( publisher_id INT PRIMARY KEY,
name VARCHAR(100), city VARCHAR(100),
country VARCHAR(100) );

CREATE TABLE CATEGORY ( category_id INT PRIMARY KEY, description VARCHAR(100) );

CREATE TABLE CATALOG ( book_id INT PRIMARY KEY,
title VARCHAR(100), author_id INT, publisher_id INT,
category_id INT, year INT, price INT,
FOREIGN KEY (author_id) REFERENCES AUTHOR(author_id),
FOREIGN KEY (publisher_id) REFERENCES PUBLISHER(publisher_id),
FOREIGN KEY (category_id) REFERENCES CATEGORY(category_id) );

CREATE TABLE ORDER_DETAILS ( order_no INT, book_id INT,
quantity INT, FOREIGN KEY (book_id) REFERENCES CATALOG(book_id) );
```

Inserting sample data:

```
INSERT INTO AUTHOR (author_id, name, city, country) VALUES
(1, 'John Doe', 'New York', 'USA'), (2, 'Jane Smith', 'London', 'UK'),
(3, 'Michael Johnson', 'Paris', 'France'),(4, 'Maria Garcia', 'Madrid', 'Spain'),
(5, 'David Lee', 'Tokyo', 'Japan');

INSERT INTO PUBLISHER (publisher_id, name, city, country) VALUES
(1, 'Publisher A', 'New York', 'USA'), (2, 'Publisher B', 'London', 'UK'),
(3, 'Publisher C', 'Paris', 'France'), (4, 'Publisher D', 'Madrid', 'Spain'),
(5, 'Publisher E', 'Tokyo', 'Japan');

INSERT INTO CATEGORY (category_id, description) VALUES
(1, 'Fiction'), (2, 'Non-Fiction'), (3, 'Science Fiction'), (4, 'Thriller'), (5, 'Mystery');

INSERT INTO CATALOG (book_id, title, author_id, publisher_id, category_id, year, price) VALUES (1,
'Book 1', 1, 1, 1, 2015, 20), (2, 'Book 2', 1, 1, 2, 2012, 25),
(3, 'Book 3', 2, 2, 3, 2018, 30), (4, 'Book 4', 3, 3, 4, 2019, 35), (5, 'Book 5', 3, 4, 5, 2013, 40);

INSERT INTO ORDER_DETAILS (order_no, book_id, quantity) VALUES (1, 1, 5), (2, 2, 3),
(3, 3, 2), (4, 4, 4), (5, 5, 6);
```

ii) Authors with 2 or more books, price greater than average, and published after 2010:

```
SELECT A.name AS author_name FROM AUTHOR A
JOIN CATALOG C ON A.author_id = C.author_id WHERE C.year > 2010
GROUP BY A.author_id
HAVING COUNT(C.book_id) >= 2 AND AVG(C.price) < (SELECT AVG(price) FROM CATALOG)
```

iii) Author of the book with maximum sales:

```
SELECT A.name AS author_name FROM AUTHOR A
JOIN CATALOG C ON A.author_id = C.author_id
JOIN ORDER_DETAILS O ON C.book_id = O.book_id
GROUP BY A.author_id ORDER BY SUM(O.quantity) DESC LIMIT 1;
```

iv) Increase price of books published by a specific publisher by 10%:

```
UPDATE CATALOG SET price = price * 1.1
WHERE publisher_id = (SELECT publisher_id FROM PUBLISHER
WHERE name = 'Publisher A');
```

7. Consider the following database for BANK.

BRANCH (branch-name: string, branch-city: string, assets: real)
ACCOUNT (accno: int, branch-name: string, balance: real)
DEPOSITOR (customer-name: string, accno: int)
CUSTOMER (customer-name: string, customer-street: string, customer-city: string)
LOAN (loan-no: int, branch-name: string, amount: real)
BORROWER (customer-name: string, loan-no: int)

i) Create the above tables by properly specifying the primary keys and foreign keys. Enter at least five tuples for each relation.
ii) Find all the customers who have at least two accounts at the main branch.
iii) Find all customers who have an account at all the branches located in a specific city.
iv) Demonstrate how to delete all account tuples at every branch located in specific city.

Solution:

i) Creating tables with primary keys and foreign keys:

```
CREATE TABLE BRANCH ( branch_name VARCHAR(100) PRIMARY KEY, branch_city VARCHAR(100),
assets REAL );

CREATE TABLE ACCOUNT ( accno INT PRIMARY KEY, branch_name VARCHAR(100),
balance REAL, FOREIGN KEY (branch_name) REFERENCES BRANCH(branch_name) );

CREATE TABLE DEPOSITOR ( customer_name VARCHAR(100), accno INT,
FOREIGN KEY (accno) REFERENCES ACCOUNT(accno) );

CREATE TABLE CUSTOMER ( customer_name VARCHAR(100) PRIMARY KEY, customer_street
VARCHAR(100), customer_city VARCHAR(100) );

CREATE TABLE LOAN ( loan_no INT PRIMARY KEY, branch_name VARCHAR(100),
amount REAL, FOREIGN KEY (branch_name) REFERENCES BRANCH(branch_name) );

CREATE TABLE BORROWER ( customer_name VARCHAR(100), loan_no INT, FOREIGN KEY
(customer_name) REFERENCES CUSTOMER(customer_name), FOREIGN KEY (loan_no) REFERENCES
LOAN(loan_no) );
```

Inserting sample data:

```
INSERT INTO BRANCH (branch_name, branch_city, assets) VALUES
('Main', 'New York', 100000), ('Downtown', 'Los Angeles', 80000),
 ('Uptown', 'Chicago', 120000), ('Westside', 'San Francisco', 90000),
('East End', 'Boston', 110000);

INSERT INTO ACCOUNT (accno, branch_name, balance) VALUES
(1, 'Main', 5000), (2, 'Main', 7000), (3, 'Main', 3000),
(4, 'Downtown', 4000), (5, 'Uptown', 6000);
```

INSERT INTO DEPOSITOR (customer_name, accno) VALUES ('John Doe', 1),
('John Doe', 2),('Jane Smith', 3), ('Jane Smith', 4), ('David Lee', 5);

INSERT INTO CUSTOMER (customer_name, customer_street, customer_city) VALUES
('John Doe', '123 Main St', 'New York'), ('Jane Smith', '456 Elm St', 'Los Angeles'),
('David Lee', '789 Oak St', 'Chicago'), ('Michael Johnson', '101 Pine St', 'San Francisco'),
('Maria Garcia', '202 Maple St', 'Boston');

INSERT INTO LOAN (loan_no, branch_name, amount) VALUES (101, 'Main', 10000),
(102, 'Downtown', 15000), (103, 'Uptown', 20000), (104, 'Westside', 12000),
(105, 'East End', 18000);

INSERT INTO BORROWER (customer_name, loan_no) VALUES ('John Doe', 101),
('Jane Smith', 102), ('David Lee', 103), ('Michael Johnson', 104), ('Maria Garcia', 105);

ii) Customers with at least two accounts at the main branch:
SELECT customer_name FROM DEPOSITOR
WHERE accno IN (SELECT accno FROM ACCOUNT WHERE branch_name = 'Main')
GROUP BY customer_name HAVING COUNT(*) >= 2;

iii) Customers who have accounts at all branches located in a specific city (e.g., New York):
SELECT customer_name FROM DEPOSITOR
WHERE accno IN (SELECT accno FROM ACCOUNT
WHERE branch_name IN (SELECT branch_name FROM BRANCH
WHERE branch_city = 'New York'))
GROUP BY customer_name HAVING COUNT(DISTINCT branch_name) = (SELECT COUNT(*) FROM
BRANCH WHERE branch_city = 'New York');

iv) Deleting all account tuples at every branch located in a specific city (e.g., Los Angeles):
DELETE FROM ACCOUNT WHERE branch_name IN (SELECT branch_name FROM BRANCH
WHERE branch_city = 'Los Angeles');

8. Consider the following database for ORDER PROCEESING.
CUSTOMER (cust-no: int, cname: string, city: string)
ORDER (orderno: int, odate: date, ord-amt: real)
ORDER_ITEM (orderno: int, itemno:int, qty: int)
ITEM (itemno: int, unit price: real)
SHIPMENT (orderno: int, warehouseno: int, ship-date: date)
WAREHOUSE (warehouseno: int, city: string)

i) Create the above tables by properly specifying the primary keys and the foreign keys. Enter at least five tuples for each relation.
ii) List the order number and ship date for all orders shipped from particular warehouse.
iii) Produce a listing: customer name, no of orders, average order amount.
iv) List the orders that were not shipped within 30 days of ordering

Solution:
i) Creating tables with primary keys and foreign keys:
CREATE TABLE CUSTOMER (cust_no INT PRIMARY KEY,
cname VARCHAR(100), city VARCHAR(100));

CREATE TABLE ORDER (orderno INT PRIMARY KEY, odate DATE, ord_amt REAL,
cust_no INT, FOREIGN KEY (cust_no) REFERENCES CUSTOMER(cust_no));

```
CREATE TABLE ORDER_ITEM ( orderno INT, itemno INT, qty INT,
PRIMARY KEY (orderno, itemno),
FOREIGN KEY (orderno) REFERENCES ORDER(orderno) );

CREATE TABLE ITEM ( itemno INT PRIMARY KEY, unit_price REAL );

CREATE TABLE SHIPMENT ( orderno INT, warehouseno INT, ship_date DATE,
PRIMARY KEY (orderno), FOREIGN KEY (orderno) REFERENCES ORDER(orderno),
FOREIGN KEY (warehouseno) REFERENCES WAREHOUSE(warehouseno) );

CREATE TABLE WAREHOUSE ( warehouseno INT PRIMARY KEY, city VARCHAR(100) );
```

Inserting sample data:

```
INSERT INTO CUSTOMER (cust_no, cname, city) VALUES
(1, 'John Doe', 'New York'), (2, 'Jane Smith', 'Los Angeles'),
(3, 'Michael Johnson', 'Chicago'), (4, 'Maria Garcia', 'Houston'), (5, 'David Lee', 'San Francisco');

INSERT INTO ORDER (orderno, odate, ord_amt, cust_no) VALUES (101, '2024-01-05', 200.50, 1),
(102, '2024-02-10', 150.75, 2), (103, '2024-03-15', 300.00, 3),
(104, '2024-04-20', 400.25, 4), (105, '2024-05-25', 250.90, 5);

INSERT INTO ORDER_ITEM (orderno, itemno, qty) VALUES (101, 1, 2),
(102, 2, 3), (103, 3, 1), (104, 4, 4), (105, 5, 2);
INSERT INTO ITEM (itemno, unit_price) VALUES (1, 25.50), (2, 10.25),
(3, 50.00), (4, 15.75), (5, 30.90);
INSERT INTO WAREHOUSE (warehouseno, city) VALUES (1, 'New York'),
(2, 'Los Angeles'), (3, 'Chicago'), (4, 'Houston'), (5, 'San Francisco');

INSERT INTO SHIPMENT (orderno, warehouseno, ship_date) VALUES
(101, 1, '2024-01-10'), (102, 2, '2024-02-15'), (103, 3, '2024-03-20'),
(104, 4, '2024-04-25'), (105, 5, '2024-06-01');
```

Shipping date is after 30 days

ii) Orders shipped from a particular warehouse:

```
SELECT orderno, ship_date FROM SHIPMENT
WHERE warehouseno = (SELECT warehouseno FROM WAREHOUSE WHERE city = 'New York');
```

iii) Listing customer name, number of orders, and average order amount:

```
SELECT C.cname AS customer_name, COUNT(O.orderno) AS num_orders, AVG(O.ord_amt) AS
avg_order_amt FROM CUSTOMER C
LEFT JOIN ORDER O ON C.cust_no = O.cust_no GROUP BY C.cname;
```

iv) Orders that were not shipped within 30 days of ordering:

```
SELECT orderno, odate, ship_date FROM ORDER
LEFT JOIN SHIPMENT ON ORDER.orderno = SHIPMENT.orderno
WHERE DATEDIFF(ship_date, odate) > 30 OR ship_date IS NULL;
```

This query assumes that the "ship_date" column is NULL if the order hasn't been shipped yet. Adjustments may be necessary based on the actual schema and data constraints.

*****