Midterm Preparation Solution

Problem 1:

Calculate the closures of all (non-trivial) subsets of attributes for the relation R(A,B,C,D) with four functional dependencies $AB \rightarrow C, \;\; AB \rightarrow D, \;\; C \rightarrow A, \;\; D \rightarrow B$. Find the keys. Determine whether R is in Third, Boyce-Codd, or Fourth Normal Form.

$\{A\}^+ = \{A\}$
$\{B\}^+ = \{B\}$
$\{C\}^+ = \{A, C\}$
$\{D\}^+ = \{B, D\}$
$\{A, B\}^+ = \{A, B, C, D\}$ so AB is a key
$\{A, C\}^+ = \{A, C\}$
$\{A, D\} = \{A, B, C, D\}$ so AD is a key
$\{B, C\}^+ = \{A, B, C, D\}$ so BC is a key
$\{B, D\}^+ = \{B, D\}$
$\{C, D\}^+ = \{A, B, C, D\}$ so CD is a key
$\{A, B, C\}^+ = \{A, B, C, D\}$ (superkey)
$\{B, C, D\}^+ = \{A, B, C, D\}$ (superkey)
$\{A, C, D\}^+ = \{A, B, C, D\}$ (superkey)
$\{A, B, D\}^+ = \{A, B, C, D\}$ (superkey)

A relation is in third normal form if whenever we have a nontrivial FD, either the right side is a superkey or the attributes on the right side of the FDs are members of some key. Notice that all attributes are parts of a key, therefore the second condition is always fulfilled. This relation is in third normal form.

A relation is in Boyce-Codd normal form, if for all nontrivial functional functional dependencies, the left side is a super-key. Since C is not a key, the functional dependency $C \rightarrow A$ violates BCNF. For the same reason, the relation is not in 4NF.

Problem 2:

```
CREATE DATABASE IF NOT EXISTS sqlsample;
USE sqlsample;

CREATE TABLE department (
    dept_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    CONSTRAINT pk_department PRIMARY KEY (dept_id)
);

CREATE TABLE branch (
    branch_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    address VARCHAR(30),
    city VARCHAR(20),
    state VARCHAR(2),
    zip VARCHAR(12),
    CONSTRAINT pk_branch PRIMARY KEY (branch_id)
);

CREATE TABLE employee (
    emp_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    fname VARCHAR(20) NOT NULL,
    lname VARCHAR(20) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE,
    superior_emp_id SMALLINT UNSIGNED,
    dept_id SMALLINT UNSIGNED,
    title VARCHAR(20),
    assigned_branch_id SMALLINT UNSIGNED,
    CONSTRAINT fk_e_emp_id FOREIGN KEY (superior_emp_id)
        REFERENCES employee (emp_id),
    CONSTRAINT fk_dept_id FOREIGN KEY (dept_id)
        REFERENCES department (dept_id),
    CONSTRAINT fk_e_branch_id FOREIGN KEY (assigned_branch_id)
        REFERENCES branch (branch_id),
    CONSTRAINT pk_employee PRIMARY KEY (emp_id)
);


CREATE TABLE product_type (
    product_type_cd VARCHAR(10) NOT NULL,
    name VARCHAR(50) NOT NULL,
    CONSTRAINT pk_product_type PRIMARY KEY (product_type_cd)
);

CREATE TABLE product (
    product_cd VARCHAR(10) NOT NULL,
    name VARCHAR(50) NOT NULL,
    product_type_cd VARCHAR(10) NOT NULL,
    date_offered DATE,
    date_retired DATE,
    CONSTRAINT fk_product_type_cd FOREIGN KEY (product_type_cd)
        REFERENCES product_type (product_type_cd),
    CONSTRAINT pk_product PRIMARY KEY (product_cd)
);
```

```sql
CREATE TABLE customer (
    cust_id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    fed_id VARCHAR(12) NOT NULL,
    cust_type_cd ENUM('I', 'B') NOT NULL,
    address VARCHAR(30),
    city VARCHAR(20),
    state VARCHAR(20),
    postal_code VARCHAR(10),
    CONSTRAINT pk_customer PRIMARY KEY (cust_id)
);


CREATE TABLE individual (
    cust_id INTEGER UNSIGNED NOT NULL,
    fname VARCHAR(30) NOT NULL,
    lname VARCHAR(30) NOT NULL,
    birth_date DATE,
    CONSTRAINT fk_i_cust_id FOREIGN KEY (cust_id)
        REFERENCES customer (cust_id),
    CONSTRAINT pk_individual PRIMARY KEY (cust_id)
);


CREATE TABLE business (
    cust_id INTEGER UNSIGNED NOT NULL,
    name VARCHAR(40) NOT NULL,
    state_id VARCHAR(10) NOT NULL,
    incorp_date DATE,
    CONSTRAINT fk_b_cust_id FOREIGN KEY (cust_id)
        REFERENCES customer (cust_id),
    CONSTRAINT pk_business PRIMARY KEY (cust_id)
);


CREATE TABLE officer (
    officer_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    cust_id INTEGER UNSIGNED NOT NULL,
    fname VARCHAR(30) NOT NULL,
    lname VARCHAR(30) NOT NULL,
    title VARCHAR(20),
    start_date DATE NOT NULL,
    end_date DATE,
    CONSTRAINT fk_o_cust_id FOREIGN KEY (cust_id)
        REFERENCES business (cust_id),
    CONSTRAINT pk_officer PRIMARY KEY (officer_id)
);


CREATE TABLE account (
    account_id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    product_cd VARCHAR(10) NOT NULL,
    cust_id INTEGER UNSIGNED NOT NULL,
    open_date DATE NOT NULL,
    close_date DATE,
    last_activity_date DATE,
    status ENUM('ACTIVE', 'CLOSED', 'FROZEN'),
    open_branch_id SMALLINT UNSIGNED,
    open_emp_id SMALLINT UNSIGNED,
    avail_balance FLOAT(10 , 2 ),
    pending_balance FLOAT(10 , 2 ),
    CONSTRAINT fk_product_cd FOREIGN KEY (product_cd)
        REFERENCES product (product_cd),
    CONSTRAINT fk_a_cust_id FOREIGN KEY (cust_id)
        REFERENCES customer (cust_id),
    CONSTRAINT fk_a_branch_id FOREIGN KEY (open_branch_id)
        REFERENCES branch (branch_id),
    CONSTRAINT fk_a_emp_id FOREIGN KEY (open_emp_id)
        REFERENCES employee (emp_id),
    CONSTRAINT pk_account PRIMARY KEY (account_id)
);


CREATE TABLE transaction (
    txn_id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    txn_date DATETIME NOT NULL,
    account_id INTEGER UNSIGNED NOT NULL,
    txn_type_cd ENUM('DBT', 'CDT'),
    amount DOUBLE(10 , 2 ) NOT NULL,
    teller_emp_id SMALLINT UNSIGNED,
    execution_branch_id SMALLINT UNSIGNED,
    funds_avail_date DATETIME,
    CONSTRAINT fk_t_account_id FOREIGN KEY (account_id)
        REFERENCES account (account_id),
    CONSTRAINT fk_teller_emp_id FOREIGN KEY (teller_emp_id)
        REFERENCES employee (emp_id),
    CONSTRAINT fk_exec_branch_id FOREIGN KEY (execution_branch_id)
        REFERENCES branch (branch_id),
    CONSTRAINT pk_transaction PRIMARY KEY (txn_id)
);


/* end table creation */
```

Problem 3:

Use the chase test to determine whether the decomposition of $R(A, B, C, D, E)$ into $S(A, B, C)$, $T(B, C, D)$ and $V(C, D, E)$ with FDs $A, B \rightarrow C$, $B \rightarrow D$, and $C, D \rightarrow E$ in R is a lossless decomposition?

Solution:

We start out with a tableau with one line for each projection:

A	B	C	D	E
a	b	c	d_1	e_1
a_2	b	c	d	e_2
a_3	b_3	c	d	e

We can only apply the FDs $B \rightarrow D$ and $C, D \rightarrow E$. The first one changes the first row in the D-column.

A	B	C	D	E
a	b	c	d	e_1
a_2	b	c	d	e_2
a_3	b_3	c	d	e

Applying $C, D \rightarrow E$ changes all the entries in the E-column since the values in the C- and D-column are all (c,d):

A	B	C	D	E
a	b	c	d	e
a_2	b	c	d	e
a_3	b_3	c	d	e

Now however we are stuck. The FD $B \rightarrow D$ does not give us anything new. The FD $C, D \rightarrow E$ was just applied. This leaves only $A, B \rightarrow C$, but the pairs in the A- and B-column have all different values.

Problem 4:

Find the names and credit limits of customers with a credit limit of 200,000 or more.

```
SELECT
    c.customerName, c.creditLimit
FROM
    customers c
WHERE
    creditLimit >= 200000;
```

Which customer made a payment of 7678.25?

```
SELECT
    customers.customerName, customers.city, customers.country
FROM
    customers,
    payments
WHERE
    customers.customerNumber = payments.customerNumber
        AND payments.amount = 7678.25;
```

Find the full names of all contacts at customers and of all employees.

```
SELECT
    c.contactFirstName AS 'First Name',
    c.contactLastName AS 'Last Name'
FROM
    customers c
UNION SELECT
    e.firstName AS 'First Name', e.lastName AS 'Last Name'
FROM
    employees e;
```

Find the names, city, and country of customers from Germany who are not in Berlin.

```
SELECT
    c.customerName, c.country, c.city
FROM
    customers c
WHERE
    c.city <> 'Berlin'
        AND c.country = ‘Germany’;
```

Find the customer name, order Number, order date and shipped date whenever the shipped date is more than 60 days after the ordered date.

```
SELECT
    c.customerName, o.orderNumber, o.orderDate, o.shippedDate
FROM
    customers c
```

```
        INNER JOIN
    orders o
WHERE
    DATEDIFF(o.shippedDate, o.orderDate) >= 60;
```

Find the product name and the quantity ordered where 70 or more items are ordered in the same order. Order the result by the quantity ordered starting with the largest order.

```
SELECT
    p.productName, od.quantityOrdered
FROM
    orderdetails od
        INNER JOIN
    products p USING (productCode)
WHERE
    od.quantityOrdered >= 70
ORDER BY od.quantityOrdered DESC;
```

Find the name, country, and city of customers who ordered a model with “Gran Torino” in the product description.

```
SELECT
    c.customerName, c.country, c.city
FROM
    customers c
        INNER JOIN
    orders o USING (customerNumber)
        INNER JOIN
    orderdetails od USING (orderNumber)
        INNER JOIN
    products p USING (productCode)
WHERE
    p.productName LIKE '%Gran Torino%'
        AND c.country = ‘Germany';
```

Find the names and countries of customers who did not make a payment in 2003.

```
SELECT
    c.customerName, c.city, c.country
FROM
    customers c
WHERE
    customerNumber NOT IN (SELECT
            customerNumber
        FROM
            payments
        WHERE
            YEAR(payments.paymentDate) = 2003);
```

Find the number of customers who never had an order cancelled.

```
SELECT
    COUNT(c.customerName)
FROM
    customers c
WHERE
    c.customerName NOT IN (SELECT
            c1.customerName
        FROM
            customers c1
                INNER JOIN
            orders USING (customerNumber)
        WHERE
            status = 'cancelled');
```