Midterm Solutions

Problem 1:

```
CREATE TABLE customer (
    customer_id INT NOT NULL AUTO_INCREMENT,
    first_name VARCHAR(63),
    last_name VARCHAR(63),
    address VARCHAR(255) DEFAULT '',
    PRIMARY KEY (customer_id)
);

CREATE TABLE complaint (
    complaint_id INT NOT NULL AUTO_INCREMENT,
    customer_id INT,
    complaint TEXT,
    PRIMARY KEY (complaint_id),
    CONSTRAINT FK_customer_id FOREIGN KEY (customer_id)
        REFERENCES customer (customer_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Problem 2

```
SELECT
    first_name, last_name
FROM
    employees
        JOIN
    salaries USING (emp_no)
WHERE
    salary = (SELECT
            MAX(salary)
        FROM
            salaries);
```

first_name	last_name
Dietmar	Brandt

Problem 3:

```
CREATE TEMPORARY TABLE top_ten_salaries
(SELECT
    salary
FROM
    salaries
ORDER BY salary DESC
LIMIT 10);

SELECT DISTINCT
    first_name, last_name
FROM
    employees
        JOIN
    salaries USING (emp_no)
WHERE
    salary IN (SELECT
```

first_name	last_name
Joseph	Carter
Arjun	Pathak
Dietmar	Brandt
Emily	Morgan

```
            *
        FROM
            top_ten_salaries);
```

Problem 4:

```
SELECT
    dep_name, AVG(salary) AS 'Average Salary'
FROM
    salaries
        JOIN
    departments ON dep_no = dep_id
WHERE
    start_day <= '2020-01-06'
        AND end_day >= '2020-01-06'
GROUP BY dep_name;
```

Problem 5:

```
SELECT
    first_name, last_name
FROM
    employees
        JOIN
    salaries USING (emp_no)
        JOIN
    departments ON dep_no = dep_id
WHERE
    office_id IN (SELECT
            office_id
        FROM
            employees
                JOIN
            salaries USING (emp_no)
                JOIN
            departments ON dep_no = dep_id
        WHERE
            employees.first_name = 'Ingrid'
                AND last_name = 'Koch')
        AND hire_day BETWEEN (SELECT
            hire_day
        FROM
            employees
        WHERE
            first_name = 'Ingrid'
                AND last_name = 'Koch') AND (SELECT
            last_day
        FROM
            employees
        WHERE
            first_name = 'Ingrid'
                AND last_name = 'Koch');
```

first_name	last_name
Ian	Iglesias
Francisco	Serrano
Sandra	Gray
Albert	Evans
Emilio	Sánchez
Lisa	Bailey
Ingrid	Koch
Jacobo	Sanz
María	Pérez
Ishani	Prakash
Valery	García
Aarya	Chaudhari
Beate	Klein
Pablo	Gómez
Anja	Lehmann
Fátima	Rubio
Marie	Thompson
Margarete	Weber
Emma	Carter
Kanha	Molla
Srihan	Debbarma
Rose	Wilson
Virat	Alam
Paula	Martín
Emilio	Marín

Problem 6:

(a) InsuranceCompName -> InsuranceCompAddress and Name, Birthday, Phone -> Address are functional dependencies (among others) where the left side is not a super-key.

(b) $\{A\}^+ = \{A, D, E\}$; $\{B\}^+ = \{B\}$; $\{C\}^+ = \{C\}$; $\{D\}^+ = \{A, D, E\}^+$; $\{E\}^+ = \{E\}$.
$\{A, B\}^+ = \{A, B, D, E\}$; $\{A, C\}^+ = \{A, C, D, E\}$; $\{A, D\}^+ = \{A, D, E\}$;
$\{A, E\} = \{A, D, E\}$; $\{B, C\}^+ = \{A, B, C\}^+ = \{A, B, C, D, E\}$, i.e. BC is a key.
$\{B, D\}^+ = \{A, D, E\}$; $\{B, E\}^+ = \{B, E\}$; $\{C, D\}^+ = \{A, C, D, E\}$;
$\{C, E\}^+ = \{C, E\}$; $\{D, E\}^+ = \{A, D, E\}$.

(c) We set up a table where each row corresponds to a decomposition table and the columns to the attributes of the original table. This gives us

A	B	C	D	E
a	b_1	c	d	e_1
a	b	c_2	d_2	e
a_3	b	c_3	d	e_3

We first apply $A \to D$. This gives

A	B	C	D	E
a	b_1	c	d	e_1
a	b	c_2	d	e
a_3	b	c_3	d	e_3

We then can apply $A \to E$. This gives

A	B	C	D	E
a	b_1	c	d	e
a	b	c_2	d	e
a_3	b	c_3	d	e_3

We can apply $D \to A$. This gives

A	B	C	D	E
a	b_1	c	d	e
a	b	c_2	d	e
a	b	c_3	d	e_3

We can apply $D \to E$. This gives

A	B	C	D	E
a	b_1	c	d	e
a	b	c_2	d	e
a	b	c_3	d	e

No more functional dependencies will change the table. We can now construct a counterexample by just using this table as relations in the original database table. This table fulfills all functional dependencies

A	B	C	D	E
a	b_1	c	d	e
a	b	c_2	d	e
a	b	c_3	d	e

and has projections

A	C	D
a	c	d
a	c_2	d
a	c_3	d

A	B	E
a	b_1	e
a	b	e

B	D
b_1	d
b	d

When we join, we get

A	B	C	D	E
a	b_1	c	d	e
a	b_1	c_2	d	e
a	b_1	c_3	d	e
a	b	c	d	e
a	b	c_2	d	e

A	B	C	D	E
a	b	c_3	d	e

which obviously has more rows than the original database table.