Part A

Q1

If B is dependent on A and C is dependent on B, there exists a transitive functional dependency A > B > C.

Q2

First normal form requires that there are no repeated entities inside a single table cell.

Q3

2NF states that a table must be in 1NF and have no partial dependencies from CKs to non-prime attributes. However, 3NF, as well as requiring 2NF, states that a table must have no transitive dependencies from CKs to non-prime attributes.

Q4

a) Inconsistency between camel case and snake case

b) animal_id should be real or serial rather than text, name should be text not real, keeper name should be text, last_fed should be timestamp, is Endangered should be Boolean

3 marks for all of them, 2 marks for 3/4, 1 mark for 1/2/3

c) Deletion errors - loss of the keeper's information

Q5

Attribute = column = field each row can have
Row = record = fact about an entity
Table = relation = set of rows and columns
Relationship = connection between tables, mediated by foreign keys
Key = attribute or set of attributes which can uniquely identify rows

Q6

a) Answer should mention insert/update errors (inconsistency if multiple copies).
b) Answer should mention insert/update errors or delete errors.

Part B

Q1

Yes, via alliances.

Q2

No, treaties uses two one-to-one relationships rather than a many-to-many relationship.

Q3

```
SELECT name, in_office_start
FROM leaders
```

Q4

```
SELECT leaders.*, COUNT(*) AS title_count
FROM leaders LEFT OUTER JOIN given_titles
ON leaders.id = given_titles.leader_id
GROUP BY leaders.id
```

Watch out for the LEFT OUTER JOIN!

Q5

```
SELECT countries.id, countries.name, MIN(start)
FROM countries INNER JOIN countries_in_alliances
ON countries.id = countries_in_alliances.country_id
GROUP BY countries.id
```

Q6

The subquery provides the first alliance dates for every country. This is then used to join to the right rows in the alliances table.

```
SELECT countries.name, first_alliances.first_alliance_date,
alliances.name
FROM countries INNER JOIN

(SELECT countries.id,
MIN(start) AS first_alliance_date
FROM countries INNER JOIN countries_in_alliances
ON countries.id = countries_in_alliances.country_id
INNER JOIN alliances
ON countries_in_alliances.alliance_id = alliances.alliance_id
GROUP BY countries.id) AS first_alliances

INNER JOIN countries_in_alliances
ON countries.id = countries_in_alliances.country_id
INNER JOIN alliances
ON countries_in_alliances.alliance_id = alliances.alliance_id
AND alliances.date = first_alliances.first_alliance_date
```

Q7

The question is no longer properly defined. In the previous query, the INNER JOIN will multiply the rows, showing two rows for each country.

Q8

```
SELECT alliances.name, SUM(population) AS total_population
FROM
alliances INNER JOIN countries_in_alliances
ON alliances.id = countries_in_alliances.alliance_id
INNER JOIN countries
ON countries_in_alliances.country_id = countries.id
GROUP BY alliances.id
```

Q9

```
SELECT * FROM leaders
WHERE in_office_start IS NOT NULL
AND in_office_end IS NULL
```

Q10

```
SELECT countries.name, COUNT(DISTINCT treaties.id) AS treaty_count
FROM
countries
INNER JOIN treaties
ON countries.id = treaties.country_1_id
OR countries.id = treaties.country_2_id
GROUP BY countries.id
```

Q11

```
SELECT(
SELECT leaders.id FROM
(
SELECT leaders.id, COUNT(*) AS title_count
FROM
leaders INNER JOIN given_titles
ON leaders.id = given_titles.leader_id
WHERE in_office_end IS NOT NULL
AND in_office_end IS NULL
GROUP BY leaders.id
ORDER BY title_count DESC
LIMIT 1
)
)
=
(
SELECT leaders.id FROM
(SELECT id FROM leaders
WHERE in_office_start IS NOT NULL
AND in_office_end IS NULL
ORDER BY in_office_start ASC
LIMIT 1)
)
```

Q12

```
SELECT * c1.name, c2.name,
COUNT(*) OVER (ORDER BY date) AS
FROM

treaties INNER JOIN countries AS c1
ON treaties.country_1_id = c1.id
INNER JOIN countries as c2
ON treaties.country_2_id = c2.id
ORDER BY date
```