I am giving you relations in the form of partially-specified CREATE TABLE statements with attribute types that are consistent with some SQL interpreters (though not others).

```
/* Observer associates computer network identifiers (e.g., IP addresses) with campus
     dormitories. DormName can be NULL, thus allowing easy recording of off-campus (or
     otherwise non-dorm) network identifiers of virtual visitors to dorm web-based electricity and
     water usage summaries. */

CREATE TABLE Observer (
        NetworkID CHAR(20),
        DormName VARCHAR(35),
        PRIMARY KEY (NetworkID)
);
```

```
/* A record of visits (from observers) to Dorms (often assembled, on demand) Webpage, which displays statistics
    on electricity and water usage */

CREATE TABLE DormWebPage (
        DWPageID INTEGER,
        CreateDate DATE NOT NULL, /* NOT NULL indicates field cannot be NULL in any record */
        CreateTime TIME NOT NULL,
        ObserverNetworkID CHAR(20) NOT NULL,    /* Corresponds to a NetworkID in Observer */
        DormName VARCHAR(35) NOT NULL,          /* Corresponds to a DormName in Dorm */
        PRIMARY KEY (DWPageID)
);
```

```
/*  */

CREATE TABLE Dorm (
        DormName VARCHAR(35),
        MaxOccupancy SMALLINT,
        PRIMARY KEY (DormName)
);



 /* A table of time-stamped outdoor temperatures (required) and light conditions (optional) */

 CREATE TABLE AmbientValues (
        AmbientReadingsDate DATE,
        AmbientReadingsTime TIME,
        AmbientTemp TINYINT NOT NULL,
        AmbientLight CHAR(2),
        PRIMARY KEY (AmbientReadingsDate, AmbientReadingsTime)
 );
```

```
/* If you wanted to allow Dorms to move into HighRes status, perhaps you would make StartDate part of the PK */

CREATE TABLE HighResDorm (
        DormName VARCHAR(35),       /* Corresponds to a DormName in Dorm */
        StartDate DATE,             /* Date at which it became a HighResDorm */
        PRIMARY KEY (DormName)
);



/* A LowRes dorm is assumed to have a unique (NOT NULL) electricity sensor, but the definition allows water
sensors to be shared across dorms (not unique) and none at all (allowed NULL) */

CREATE TABLE LowResDorm (
        DormName VARCHAR(35),           /* Corresponds to a DormName in Dorm */
        StartDate DATE,                 /* Date at which it became a LowResDorm */
        LRElecSensorID INTEGER NOT NULL,
        UNIQUE(LRElecSensorID),  /* UNIQUE indicates a key; typically implies NOT NULL */
        LRElecSensorOnLineDate DATE,
        LRWaterSensorOnLineDate DATE,
        LRWaterSensorID INTEGER,
        PRIMARY KEY (DormName)
);
```

```
/* For example, FloorNum = 3 and DormName = McGill is 3rd floor of McGill */

CREATE TABLE Floor (
        DormName VARCHAR(35),  /* Corresponds to a DormName in HighResDorm */
        FloorNum TINYINT,
        MaxOccupancy SMALLINT,
        PRIMARY KEY (DormName, FloorNum)
);
```

```
/* Similar meanings as those of DormWebPage */

CREATE TABLE FloorWebPage (
        DormName VARCHAR(35),    /* Corresponds to DormName in Floor */
        FloorNum TINYINT,        /* Corresponds to FloorNum in Floor */
        FWPageID INTEGER,
        CreateDate DATE NOT NULL,
        CreateTime TIME NOT NULL,
        ObserverNetworkID CHAR(20) NOT NULL,   /* Corresponds to NetworkID in Observer */
        PRIMARY KEY (FWPageID)
);
```

```
/* Definition allows multiple sensors per floor (thus, DormName,Floor not required UNIQUE) */

CREATE TABLE HRElecSensor (
        DormName VARCHAR(35) NOT NULL, /* Corresponds to DormName in Floor */
        FloorNum TINYINT NOT NULL,          /* Corresponds to FloorNum in Floor */
        HRElecSensorID INTEGER,
        HRElecSensorOnLineDate DATE,
        PRIMARY KEY (HRElecSensorID)
);




/If you bother to record a reading, the value should be NOT NULL (perhaps coupled
  special value(e.g., -999)  indicating not read because not functional sensor */

CREATE TABLE HREReading (
        HRElecSensorID INTEGER,   /* Corresponds to HRElecSensorID in HRElecSensor */
        HREReadingDate DATE,
        HREReadingTime TIME,
        HREValue INTEGER NOT NULL,
        PRIMARY KEY (HRElecSensorID, HREReadingDate, HREReadingTime),
);
```

```
/* Comment here */

CREATE TABLE HRWSensor (
        DormName VARCHAR(35) NOT NULL,  /* Corresponds to DormName in Floor */
        FloorNum TINYINT NOT NULL,      /* Corresponds to FloorNum in Floor */
        HRWaterSensorID INTEGER,
        HRWaterSensorOnLineDate DATE,
        PRIMARY KEY (HRWaterSensorID)
);
```

```
/* Comment here */

CREATE TABLE HRWReading (
        HRWaterSensorID INTEGER,  /* Corresponds to HRWaterSensorID in HRWaterSensor */
        HRWReadingDate DATE,
        HRWReadingTime TIME,
        HRWValue INTEGER NOT NULL,
        PRIMARY KEY (HRWaterSensorID, HRWReadingDate, HRWReadingTime)
);
```

```
/* Comment here */

CREATE TABLE LREReading (
        DormName VARCHAR(35),   /* Corresponds to DormName in LowResDorm */
        LREReadingDate DATE,
        LREReadingTime TIME,
        LREValue INTEGER NOT NULL,
        PRIMARY KEY (DormName, LREReadingDate, LREReadingTime)
);
```

```
/* Comment here */

CREATE TABLE LRWReading (
        DormName VARCHAR(35),   /* Corresponds to DormName in LowResDorm */
        LRWReadingDate DATE,
        LRWReadingTime TIME,
        LRWValue INTEGER NOT NULL,
        PRIMARY KEY (DormName, LRWReadingDate, LRWReadingTime)
);
```

1) Write SQL queries specified below in a manner that is consistent with the table definitions.

a) Write a SQL query that lists all of the individual water readings recorded for each floor of McGill, a high res dorm, on Jan 28, 2012 (HRWReadingDate = '2012-01-28'). Each row of the output should list dorm name (all McGill), floor number, the date (yes, the dates in each row should be the same), the time of reading, and the reading value.

b) Write a SQL query that lists all of the individual water readings recorded for each floor of each high res dorm on Jan 28, 2012 (HRWReadingDate = '2012-01-28'). Each row of the output should list dorm name, floor number, the date (yes, the dates in each row should be the same), the time of reading, and the reading value.

c) Much like (b), but rather than listing all the readings for a given day, list the AVERAGE water reading values for each day, together with the date, the dorm name, and the floor number.

d) Just like (c), but list only those daily averages that are computed over more than 5 values.

e) Write an SQL query that lists, for each dorm **(low or high res)**, the AVERAGE ambient (outside) temperature and the AVERAGE electricity readings on each day in which the MINIMUM ambient temperature exceeds 70 degrees. Thus, the output should list dorm name, average electrical reading (listed as AveElec), and average ambient temperature (as AveTemp), and the reading date.

f) Write an SQL query that lists the average electrical readings for each floor that were viewed by an observer with 'FloorWebPage.ObserverNetworkID = X', for each day in which the MAX ambient temperature was greater than 80 degrees (AmbientTemp > 80). Include 'FloorWebPage.ObserverNetworkID = X' as is. Note that 'X' is a variable; when an unbound variable appears in a query, it is a parameter that is requested at interpretation time). Each row of the output should list network id (which should be the same across all rows), the AVERAGE ambient temperature for the day (as AveTemp), the dorm name, floor, sensor id, electrical reading date, and average value (as AveElec).

2) Reflect on these questions. You do NOT need to answer and submit them in writing, but they will be topics of discussion.

a) If you try to delete a row in Observer that has one or more 'associated' entries in DormWebPage, what will happen?

b) If you try to delete a Dorm, for which no one has ever looked at (Observed) a DormWebPage for it, what will happen?

c) If you try to delete a Dorm, for which there have been one or more recorded views of DormWebPages for it, what will happen?

d) How many electricity sensors are associated with a low res dorm (so far as the DB encodes)? And vice versa?

e) How many electricity sensors are associated with a high res dorm (so far as the DB encodes)?

f) Could the current database design (tables and assertions) support the situation of a low res dorm becoming a high res dorm WITHOUT losing past data from its low res days? If so, explain, and if not, explain what changes to the DB design you might make to support this (high likelihood eventuality) PRIOR to DB implementation.

g) How could the DB design be changed to support records of room and plug level measurements of electricity (and perhaps faucet level water readings)?