Download the Sales database as a single “.sql” file with both the schemas and the instances from this URL:

http://pages.cs.wisc.edu/~arun/cs564_fall15/downloads/Sales.sql

To load the database, simply type “.read Sales.sql” on the SQLite3 command prompt. Recall that the normalized sales database has the following 4 tables. The key of each table is underlined and the foreign keys are also mentioned:

- **Holidays** (WeekDate, IsHoliday)
- **Stores** (Store, Type, Size)
- **TemporalData** (Store, WeekDate, Temperature, FuelPrice, CPI, UnemploymentRate)
  - Store is a foreign key referencing Stores (Store).
  - WeekDate is a foreign key referencing Holidays (WeekDate).
- **Sales** (Store, Dept, WeekDate, WeeklySales)
  - Store is a foreign key referencing Stores (Store).
  - WeekDate is a foreign key referencing Holidays (WeekDate).
  - (Store, WeekDate) is a foreign key referencing TemporalData (Store, WeekDate).

Write SQL queries over the given normalized sales database that obtain the answers to the following questions:

1. **[5%]** Which stores had the largest and smallest overall sales during holiday weeks?
   **Answer:**

   ```sql
   DROP VIEW IF EXISTS HolSales;
   CREATE VIEW HolSales AS
   SELECT S.Store, SUM (S.WeeklySales) AS AllSales
   FROM Sales S, Holidays H
   WHERE S.WeekDate = H.WeekDate AND H.IsHoliday = "TRUE"
   GROUP BY S.Store;
   
   SELECT Store, AllSales
   FROM HolSales
   WHERE AllSales = (SELECT MAX (AllSales) FROM HolSales) OR
            AllSales = (SELECT MIN (AllSales) FROM HolSales);
   ```
2. [5%] Get the top 10 departments overall ranked by total sales normalized by the size of the store where the sales were recorded.

   Answer:

   ```sql
   SELECT S.Dept, SUM (S.WeeklySales / R.Size) AS AllSales
   FROM Sales S, Stores R
   WHERE S.Store = R.Store
   GROUP BY S.Dept
   ORDER BY AllSales DESC
   LIMIT 10;
   ```

3. [5%] Get the stores at locations where the unemployment rate exceeded 10% at least once but the fuel price never exceeded 4.0.

   Answer:

   ```sql
   SELECT DISTINCT S.Store
   FROM Sales S, TemporalData T
   WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate AND
   T.UnemploymentRate > 10
   EXCEPT
   SELECT DISTINCT S.Store
   FROM Sales S, TemporalData T
   WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate AND
   T.FuelPrice > 4;
   ```

4. [5%] How many non-holiday weeks had larger sales than the overall average sales during holiday weeks?

   Answer:

   ```sql
   DROP VIEW IF EXISTS HolSales;
   CREATE VIEW HolSales AS
   SELECT H.WeekDate, H.IsHoliday,
   SUM (S.WeeklySales) AS AllSales
   FROM Sales S, Holidays H
   WHERE S.WeekDate = H.WeekDate
   GROUP BY H.WeekDate, H.IsHoliday;

   SELECT COUNT(*)
   FROM HolSales
   WHERE IsHoliday = "FALSE" AND
   AllSales > (SELECT AVG (AllSales)
   FROM HolSales
   WHERE IsHoliday = "TRUE");
   ```
5. [5%] Get the total sales per month overall for each type of store. Since SQLite3 does not support native operations on the DATE datatype, use the LIKE predicate and the string concatenation operator ("||") of SQLite3 to create a workaround.

Answer:

```
SELECT R.Type, substr (S.WeekDate, 6, 2) AS Month,
       SUM (S.WeeklySales)
FROM Stores R, Sales S
WHERE R.Store = S.Store
GROUP BY R.Type, Month;
```

Aliter:

```
DROP TABLE IF EXISTS Months;
CREATE TABLE Months (Month CHAR(2));
INSERT INTO Months VALUES ('01'), ('02'), ('03'), ('04'), ('05'),
                         ('06'), ('07'), ('08'), ('09'), ('10'), ('11'), ('12');

SELECT R.Type, M.Month, SUM (S.WeeklySales)
FROM Stores R, Months M, Sales S
WHERE S.WeekDate LIKE "%-" || M.Month || "-%" AND
      R.Store = S.Store
GROUP BY R.Type, M.Month;
```

6. [10%] Which stores have had sales in every department in that store for every month of at least one calendar year among 2010, 2011, and 2012?

Answer:

```
DROP TABLE IF EXISTS Months;
CREATE TABLE Months (Month CHAR(2));
INSERT INTO Months VALUES ('01'), ('02'), ('03'), ('04'), ('05'),
                         ('06'), ('07'), ('08'), ('09'), ('10'), ('11'), ('12');

DROP TABLE IF EXISTS StoreDeptYM;
CREATE TABLE StoreDeptYM AS
    SELECT DISTINCT S.Store, substr (S.WeekDate, 0, 5) AS Year,
               substr (S.WeekDate, 6, 2) AS Month, S.Dept
    FROM Sales S;

DROP VIEW LoserStores;
CREATE VIEW LoserStores AS
```
7. [15%] For each of the 4 numeric attributes in TemporalData, are they positively or negatively correlated with sales? For our purposes, the intuitive notion of “correlation” is defined using a standard statistical quantity known as the Pearson correlation coefficient. Given two numeric random variables $X$ and $Y$, it is defined as follows:

$$
\rho_{X,Y} = \frac{E[XY] - E[X]E[Y]}{\sqrt{E[X^2] - E[X]^2} \sqrt{E[Y^2] - E[Y]^2}}
$$

On a given sample of data with $n$ examples each for $X$ and $Y$ (label them $x_i$ and $y_i$
respectively for \( i = 1 \ldots n \), it is estimated as follows:

\[
\rho_{X,Y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}
\]

In the above, \( \bar{x} \) and \( \bar{y} \) are the sample means, i.e., \( \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \), and \( \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \).

Your SQL query should output an instance with the following schema: Output\( \text{7} \) (AttributeName VARCHAR(20), CorrelationSign INTEGER) and 4 tuples, e.g.,\{"CPI", 1\}, \("\text{Temperature}\", -1\), \(\ldots\). In your query, the values of AttributeName can be hardcoded string literals, but the values of CorrelationSign must be computed automatically using SQL queries over the given database instance.

**Answer:**

DROP TABLE IF EXISTS AvgSales;
CREATE TABLE AvgSales AS SELECT AVG (WeeklySales) FROM Sales;

DROP TABLE IF EXISTS AvgAll;
CREATE TABLE AvgAll AS
    SELECT AVG (S.WeeklySales) AS AvgSales,
           AVG (T.Temperature) AS AvgTemp,
           AVG (T.FuelPrice) AS AvgFuel,
           AVG (T.CPI) AS AvgCPI,
           AVG (T.UnemploymentRate) AS AvgUnemp
    FROM Sales S, TemporalData T
    WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate;

DROP TABLE IF EXISTS Output;
CREATE TABLE Output (AttributeName VARCHAR (20),
                   CorrelationSign INTEGER);

INSERT INTO Output VALUES ("Temperature", 0),
                        ("FuelPrice", 0), ("CPI", 0), ("UnemploymentRate", 0);

UPDATE Output SET CorrelationSign = (SELECT SUM ((S.WeeklySales - (SELECT AvgSales FROM AvgAll))
                                           * (T.Temperature - (SELECT AvgTemp FROM AvgAll)))
                                       FROM Sales S, TemporalData T
                                       WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate)
                                       WHERE AttributeName = "Temperature";

UPDATE Output SET CorrelationSign = (SELECT SUM ((S.WeeklySales - (SELECT AvgSales FROM AvgAll))
                                           * (T.FuelPrice - (SELECT AvgFuel FROM AvgAll)))
                                       FROM Sales S, TemporalData T
                                       WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate)
                                       WHERE AttributeName = "FuelPrice";

UPDATE Output SET CorrelationSign = (SELECT SUM ((S.WeeklySales - (SELECT AvgSales FROM AvgAll))
                                           * (T.CPI - (SELECT AvgCPI FROM AvgAll)))
                                       FROM Sales S, TemporalData T
                                       WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate)
                                       WHERE AttributeName = "CPI";

UPDATE Output SET CorrelationSign = (SELECT SUM ((S.WeeklySales - (SELECT AvgSales FROM AvgAll))
                                           * (T.UnemploymentRate - (SELECT AvgUnemp FROM AvgAll)))
                                       FROM Sales S, TemporalData T
                                       WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate)
                                       WHERE AttributeName = "UnemploymentRate";
FROM Sales S, TemporalData T
WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate)
WHERE AttributeName = "FuelPrice";

UPDATE Output SET CorrelationSign = (  
  SELECT SUM ((S.WeeklySales - (SELECT AvgSales FROM AvgAll))  
    * (T.CPI - (SELECT AvgCPI FROM AvgAll))))  
FROM Sales S, TemporalData T  
WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate)  
WHERE AttributeName = "CPI";

UPDATE Output SET CorrelationSign = (  
  SELECT SUM ((S.WeeklySales - (SELECT AvgSales FROM AvgAll))  
    * (T.UnemploymentRate - (SELECT AvgUnemp FROM AvgAll))))  
FROM Sales S, TemporalData T  
WHERE S.Store = T.Store AND S.WeekDate = T.WeekDate)  
WHERE AttributeName = "UnemploymentRate";

UPDATE Output SET CorrelationSign =  
  ABS (CorrelationSign) / CorrelationSign;

SELECT * FROM Output;

PART B: THE IMDB DATABASE [50%]

Download the IMDB database as a zipped folder from this URL:

http://pages.cs.wisc.edu/~paris/cs564-f15/imdb.tar.gz

To load the database, simply type ".read create-imdb.sql" on the SQLite3 command prompt. The IMDB database has the following schema:

- **actor** (id, fname, lname, gender)
- **movie** (id, name, year)
- **directors** (id, fname, lname)
- **genre** (mid, genre)
- **casts** (pid, mid, role)
  pid is a foreign key referencing actor (id).
  mid is a foreign key referencing movie (id).
• **movie_directors** (did, mid)
  did is a foreign key referencing directors (id).
  mid is a foreign key referencing movie (id).

[15%] Before you attempt to write any SQL queries, you will create the necessary indexes to speed up the query execution. (It will be helpful to study the queries before you attempt to create any index.) To create and index, use the following statement:

```sql
CREATE [UNIQUE] INDEX index_name ON table_name(col_1, col_2, ...);
```

You can choose up to 10 indexes in total for the database. You have some flexibility about which indexes to choose, but be sure to create only indexes that matter!

**Answer:**

```
CREATE UNIQUE INDEX movieid ON movie(id);
CREATE UNIQUE INDEX actorid ON actor(id);
CREATE UNIQUE INDEX directorsid ON directors(id);
CREATE INDEX castsmid ON casts(mid);
CREATE INDEX castspid ON casts(pid);
CREATE INDEX moviefyear ON movie(year);
CREATE INDEX moviedirectorsmid ON movie_directors(mid);
CREATE INDEX moviedirectordid ON movie_directors(did);
```

After you have created the indexes, write SQL queries to answer the following questions:

1. **[5%]** List all actors (their first and last name) who have played in at least 10 different movies in 2004.
   **Answer:**
   ```sql
   SELECT a.fname, a.lname
   FROM actor a, casts c, movie m
   WHERE a.id = c.pid
     AND c.mid = m.id
     AND m.year = 2004
   GROUP BY a.id, a.fname, a.lname
   HAVING COUNT(DISTINCT m.id) > 9 ;
   ```

2. **[5%]** List the top 100 directors who have directed the most movies from 1990 to 2010, in descending order of the number of movies they have directed. Output their first name, last name, and number of movies directed.
   **Answer:**
   ```sql
   ```
SELECT d.fname, d.lname, COUNT(DISTINCT m.id) AS movie_count
FROM movie m, movie_directors md, directors d
WHERE m.id = md.mid
    AND md.did = d.id
    AND m.year >= 1990 AND m.year <= 2010
GROUP BY d.id, d.fname, d.lname
ORDER BY movie_count DESC
LIMIT 100 ;

3. [5%] For each year, count the number of movies in that year that had only female actors. Be careful: a movie without any actors is a movie that has only female actors.
   Answer:

   SELECT m.year, COUNT(*)
   FROM movie m
   WHERE m.year NOT EXISTS (SELECT *
       FROM actor a, casts c
       WHERE a.id = c.pid
           AND c.mid = m.id
           AND a.gender!='F')
   GROUP BY m.year;

4. [10%] A decade is any sequence of 10 consecutive years (e.g., 1964, 1965, ..., 1973 is a decade). Find the decade with the largest number of films (output only the first year of the decade).
   Answer:

   WITH Decade(year, movie_count) AS
   (  SELECT y.year, COUNT(*)
       FROM (SELECT DISTINCT m.year FROM movie m) y, movie m
       WHERE y.year <= m.year AND m.year < y.year+10
       GROUP BY y.year
   )
   SELECT d.year, d.movie_count
   FROM Decade d
   WHERE d.movie_count =
       (SELECT MAX(d.movie_count) FROM Decade d) ;

5. [10%] The Bacon number of an actor is the length of the shortest path between the actor and Kevin Bacon in the "co-acting" graph. Kevin Bacon has Bacon number 0;
all actors who acted in the same film as Kevin Bacon have Bacon number 1; all actors who acted in the same film as some actor with Bacon number 1 (but not with Bacon himself) have Bacon number 2, and so on. How many actors have Bacon number 2?

Answer:

```
SELECT COUNT(DISTINCT c2.pid)
FROM Actor a0, Casts c0, Casts c1a, Casts c1b, Casts c2
WHERE a0.fname = 'Kevin' AND a0.lname = 'Bacon'
  AND a0.id = c0.pid
  AND c0.mid = c1a.mid
  AND c1b.pid = c1a.pid
  AND c1b.mid = c2.mid
  AND c2.pid NOT IN
    (SELECT d1.pid
     FROM Actor b0, Casts d0, Casts d1
     WHERE b0.fname = 'Kevin' AND b0.lname = 'Bacon'
       AND b0.id = d0.pid
       AND d0.mid = d1.mid
    );
```

**Deliverables**

You can download and install SQLite3 from the following link (the current version of SQLite3 is 3.8.11.1):

[https://www.sqlite.org/download.html](https://www.sqlite.org/download.html)

You are required to submit a zipped folder with the following contents:

1. A “.sql” file per question. Name your files as “query<part><number>.sql”, e.g., the file “queryA2.sql” is for question 2 of part A.

2. A file `create-indexes.sql` that contains the SQLite commands that create the indexes for Part B

3. A text file `readme.txt` where you describe the reasoning behind the creation of the each index (e.g., this index speeds up the following query), and also report the average running time (over 3 runs) for each query with and without the indexes.