Laboratory Exercise №1

What’s inside our database

Introduction to the problem

A database most often contains one or more tables. Each database has some “schema”—the tables and columns into which the data is organized. Each table is identified by a name (e.g. "Faculties" or "Students"), and contain records (rows) with data. **Structured Query Language (SQL, usually pronounced S-Q-L or "sequel")** is a programming language designed for managing data within a Database Management System (DBMS) — a software that stores, manipulates, and retrieves data efficiently. **The SELECT statement is one of the most commonly used statements in SQL.** The SQLite SELECT statement provides all features of the SELECT statement in SQL standard. We often use the SELECT statement to query data from one or more tables.

Here’s the syntax of a possible SELECT statement:

```
SELECT DISTINCT column_list
FROM table_list
  JOIN table ON join_condition
WHERE row_filter
ORDER BY column
LIMIT count OFFSET offset
GROUP BY column
HAVING group_filter;
```

SQL keywords **are NOT case sensitive**, so **select** is the same as **SELECT**. Some database systems **require a semicolon** at the end of each SQL statement. Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

Some of the most important commands used with select clause are:

- `SELECT` - extracts data from a database
- `UPDATE` - updates data in a database
- `DELETE` - deletes data from a database
- `INSERT INTO` - inserts new data into a database
- `CREATE DATABASE` - creates a new database
- `ALTER DATABASE` - modifies a database
- `CREATE TABLE` - creates a new table
- `ALTER TABLE` - modifies a table
- `DROP TABLE` - deletes a table
- `CREATE INDEX` - creates an index (search key)
- `DROP INDEX` - deletes an index

You may find helpful the following online sources of information about SQL syntax:

- SQL Tutorial, https://www.w3schools.com/sql/default.asp
- SQL Tutorial, https://www.sqltutorial.org/

Requirements for documentation of the work done by students

Each student must prepare and send a text document (in DOCX or PDF format) with designed SQL queries for each question from tasks of the current laboratory exercise. Please read carefully the text of the questions, because **each query should return only the data necessary to answer the question! Prepared document must be sent on the e-mail of your teaching assistant no later than a day before your next laboratory exercise.**

Introduction to Task 1. Explore the information about **Cyberchase** - an animated, educational kid’s television series, aired by the United States’ Public Broadcasting Service (PBS) since 2002. In a database called **cyberchase.db**, using a table called **episodes**, students can find information about Cyberchase’s episodes thus far. Each database has some “schema”—the tables and columns into which the data is organized. In `cyberchase.db` you’ll find a single table, `episodes`. In the `episodes` table, you’ll find the following columns:

- `id`, which uniquely identifies each row (episode) in the table
- `season`, which is the season number in which the episode aired
- `episode_in_season`, which is the episode’s number within its given season
- `title`, which is the episode’s title
- `topic`, which identifies the ideas the episode aimed to teach
- `air_date`, which is the date (expressed as `YYYY-MM-DD`) on which the episode “aired” (i.e., was published)
- `production_code`, which is the unique ID used by PBS to refer to each episode internally

Task 1. For each of the following questions from the bullet list below, you should **write a single SQL query** that outputs the results specified by each problem. **Your response must take the form of a single SQL query**. You **should not** assume anything about the `id`s of any particular episodes: your queries should be accurate even if the `id` of any particular episode were different. **Finally, each query should return only the data necessary to answer the question**: if the problem only asks you to output the names of episodes, for example, then your query should not also output each episodes’s air date.

- Write a SQL query to list the titles of all episodes in *Cyberchase*’s original season, Season 1.
- List the season number of, and title of, the first episode of every season.
- Find the production code for the episode “Hackerized!”.
- Write a query to find the titles of episodes that do not yet have a listed topic.
- Find the title of the holiday episode that aired on December 31st, 2004.
- List the titles of episodes from season 6 (2008) that were released early, in 2007.
- Write a SQL query to list the titles and topics of all episodes teaching fractions.
- Write a query that counts the number of episodes released in the last 6 years, from 2018 to 2023, inclusive. You might find it helpful to know you can use `BETWEEN` with dates, such as `BETWEEN '2000-01-01' AND '2000-12-31'`.
- Write a query that counts the number of episodes released in Cyberchase’s first 6 years, from 2002 to 2007, inclusive.
- Write a SQL query to list the ids, titles, and production codes of all episodes. Order the results by production code, from earliest to latest.
- List the titles of episodes from season 5, in reverse alphabetical order.
- Count the number of unique episode titles.

- Write a SQL query to explore a question of your choice. This query should: Involve at least one condition, using `WHERE` with `AND` or `OR`.

Introduction to Task 2. From 1830 to 1832, the Japanese artist Katsushika Hokusai created 36 woodblock prints depicting 36 different views of Mount Fuji, a volcano on the Honshū island of Japan. In `views.db`, you'll find details on the 36 prints created, respectively, by Hokusai and Hiroshige. In total, you'll have data on 72 prints. In addition to each print's title and author, you'll find some statistics commonly used in computational image analysis: the print's average color, its brightness, its contrast, and its entropy.

The database `views.db` contain a single table, `views`. In the `views` table, you'll find the following columns:

- `id`, which uniquely identifies each row (print) in the table
- `print_number`, which identifies the number of the print in either Hokusai's or Hiroshige's series
- `english_title`, which is the English title of the print
- `japanese_title`, which is the Japanese title of the print
- `artist`, which is the last name of the print's artist
- `average_color`, which is the hexadecimal representation of the color found by averaging the colors of each pixel in the image
- `brightness`, which is a number corresponding to the overall lightness or darkness of the image
- `contrast`, which is a number representing the extent of the difference between light and dark areas of the image
- `entropy`, which is a measure used to quantify the complexity of the artwork

Task 2. For each of the following questions, you should write a single SQL query that outputs the results specified by each problem. Your response must take the form of a single SQL query. You **should not** assume anything about the `id`s of any particular observations: your queries should be accurate even if the `id` of any particular observation were different. Finally, each query should return only the data necessary to answer the question.

- Write a SQL query that a translator might take interest in: list, side by side, the Japanese title and the English title for each print. Ensure the Japanese title is the first column, followed by the English title.
- Write a SQL query to list the average colors of prints by *Hokusai* that include “river” in the English title. (As an aside, do they have any hint of blue?)
- Write a SQL query to count how many prints by *Hokusai* include “Fuji” in the English title. Though all of Hokusai's prints focused on Mt. Fuji, in how many did “Fuji” make it into the title?
- Write a SQL query to count how many prints by *Hiroshige* have English titles that refer to the “Eastern Capital”. Hiroshige's prints were created in Japan's “Edo period,” referencing the eastern capital city of Edo, now Tokyo.
- Write a SQL query to find the highest contrast value of prints by *Hokusai*. Name the column “Maximum Contrast”. Does Hokusai's prints most contrasting print actually have much contrast?
- Write a SQL query to find the average entropy of prints by *Hiroshige*, rounded to two decimal places. Call the resulting column “Hiroshige Average Entropy”.
- Write a SQL query to list the English titles of the 5 brightest prints by *Hiroshige*, from most to least bright.

- Write a SQL query to list the English titles of the 5 prints with the least contrast by *Hokusai*, from least to highest contrast.
- Write a SQL query to find the English title and artist of the print with the highest brightness.
- Write a SQL query to answer a question of your choice about the prints. The query should:
  - Make use of `AS` to rename a column
  - Involve at least one condition, using `WHERE`
  - Sort by at least one column, using `ORDER BY`

You can check visually results returned from your queries about brightness and contrast if you look at the original painting of the artists, available at:

- **Thirty-six Views of Mount Fuji,** https://en.wikipedia.org/wiki/Thirty-six_Views_of_Mount_Fuji
- **Thirty-six Views of Mount Fuji (Hiroshige),** https://en.wikipedia.org/wiki/Thirty-six_Views_of_Mount_Fuji_(Hiroshige)

Introduction to Task 3. How do we know whether ocean temperatures are lower or higher than “normal”? What’s a “normal” temperature? Turns out that scientists have developed a metric called a “Climate Normal.” A Climate Normal characterizes aspects of earth’s climate—its long-term weather—over a span of 30 years. One important metric is ocean temperature. In a database called `normals.db`, using a table called `normals`, explore some of the most recent Climate Normal data to understand what makes a normal ocean temperature around the world. In `normals.db` you’ll find a single table of coordinates, `normals`. In the `normals` table, you’ll find the following columns:

- `id`, which uniquely identifies each row (coordinate) in the table
- `latitude`, which is the degree of latitude (expressed in decimal format) for the coordinate
- `longitude`, which is the degree of longitude (expressed in decimal format) for the coordinate
- `0m`, which is the normal ocean surface temperature (i.e., the normal temperature at 0 meters of depth), in degrees Celsius, at the coordinate
- `5m`, which is the normal ocean temperature at 5 meters of depth, in degrees Celsius, at the coordinate
- `10m`, which is the normal ocean temperature at 10 meters of depth, in degrees Celsius, at the coordinate

And observations continue until 5500m, or 5500 meters of depth, for some coordinates!

Task 3. For each of the following questions, you should write a single SQL query that outputs the results specified by each problem. Your response must take the form of a single SQL query. You **should not** assume anything about the `id`s of any particular observations: your queries should be accurate even if the `id` of any particular observation were different. Finally, each query should return only the data necessary to answer the question.

- Write a SQL query to find the normal ocean surface temperature in the Gulf of Maine, off the coast of Massachusetts. To find this temperature, look at the data associated with **42.5° of latitude and -69.5° of longitude**. **Recall that you can find the normal ocean surface temperature in the `0m` column**, which stands for 0 meters of depth!

- Write a SQL query to find the normal temperature of the deepest sensor near the Gulf of Maine, at the same coordinate above. The deepest sensor records temperatures at 225 meters of depth, so you can find this data in the `225m` column.
- Choose a location of your own and write a SQL query to find the normal temperature at 0 meters, 100 meters, and 200 meters. You might find Google Earth (https://earth.google.com/) helpful if you’d like to find some coordinates to use!
- Write a SQL query to find the lowest normal ocean surface temperature.
- Write a SQL query to find the highest normal ocean surface temperature.
- Write a SQL query to return all normal ocean temperatures at 50m of depth, as well as their respective degrees of latitude and longitude, within the Arabian Sea (https://en.wikipedia.org/wiki/Arabian_Sea). Include latitude, longitude, and temperature columns. **For simplicity, assume the Arabian Sea is encased in the following four coordinates:**
  - 20° of latitude, 55° of longitude
  - 20° of latitude, 75° of longitude
  - 0° of latitude, 55° degrees of longitude
  - 0° of latitude, 75° degrees of longitude
- Write a SQL query to find the average ocean surface temperature, rounded to two decimal places, along the equator (https://en.wikipedia.org/wiki/Equator). Call the resulting column “Average Equator Ocean Surface Temperature”. The equator’s ocean surface temperatures can be found at all longitudes between the latitudes -0.5° and 0.5°, inclusive.
- Write a SQL query to find the 10 locations with the lowest normal ocean surface temperature, sorted coldest to warmest. If two locations have the same normal ocean surface temperature, sort by latitude, smallest to largest. Include latitude, longitude, and surface temperature columns.
- Write a SQL query to find the 10 locations with the highest normal ocean surface temperature, sorted warmest to coldest. If two locations have the same normal ocean surface temperature, sort by latitude, smallest to largest. Include latitude, longitude, and surface temperature columns.
- There are 180 whole degrees of latitude. Write a SQL query to determine how many points of latitude we have at least one data point for. (Why might we not have data points for all latitudes?)