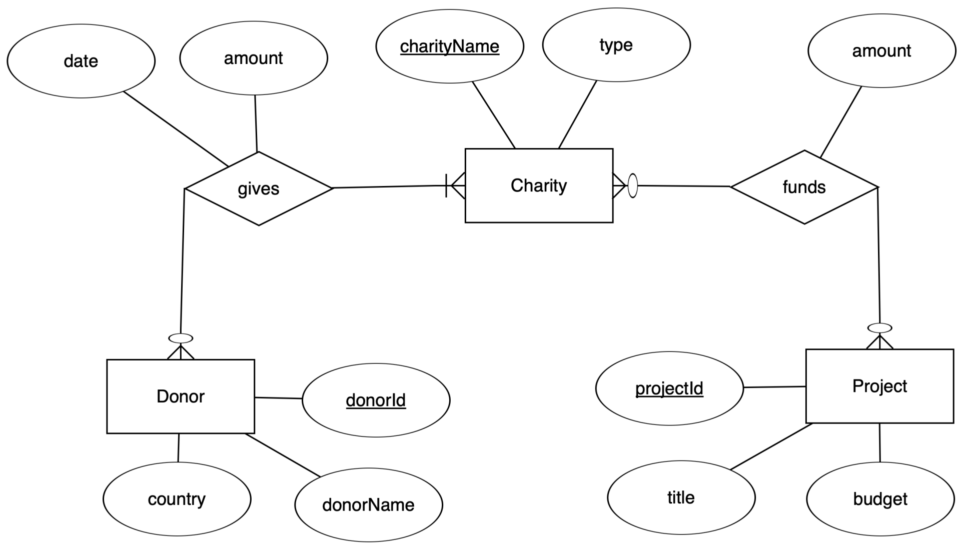
Charity Schema

The remaining questions involve writing SQLite queries for a database that records personal donations to charities which fund various projects.

The ERD for this database is as follows:



There are three entity sets `Donor`, `Charity` and `Project`

- A *donor* has a unique `donorId`, a `name` and a `country` (using the 3-letter country codes). A sample tuple in `Donor` might be `(1020, 'Bill Gates', 'USA')`.
- A *charity* has a unique `charityName` and is a particular `type` of charity, where the type is one of the values `'medical'`, `'welfare'`, `'animal'` or `'research'`, depending the main areas supported by the charity. A sample tuple from `Charity` might be `('RSPCA', 'animal')` indicating that the RSPCA is a charity dedicated to helping animals.
- A *project* has a unique `projectId`, a `title` and a `budget`. A sample tuple from `Project` might be `('X122','Novel detection of gravity waves',1000000)`.

There are two relationships, namely `gives` and `funds`, which are represented in the database by tables with the following schemas.

```
gives(donorId, charityName, amount, date)
funds(charityName, projectId, amount)
```

The table `gives` contains information about donations made to the charities. Each row represents a single donation to a particular charity. The columns `donorId` and `charityName` identify the donor and the charity respectively. The columns `amount` and `date` record the amount that has been donated and the date (given as a string in the format `'YYYY-MM-DD'`) on which the donation was made.

For example the tuple

```
(1020,'RSPCA',500000,'2022-04-01')
```

records a donation of $500k made by Bill Gates (he has donor id 1020) to the RSPCA on 1st April 2022.

The table `funds` contains information about which charities have committed funds to which projects. The fields `charityName` and `projectId` identify the charity and the project, and `amount` records how much money the charity will contribute.

```
('ARC','X122',750000)
```

records that the ARC contributed $750k to the gravity wave project.

A donor may make multiple donations to the same charity, but a project only receives funding once from a particular charity.

Usually, a project will require funding from multiple different charities to meet its entire budget, and sometimes the total funding currently awarded will be less than the project budget.

In the following questions, you will often be asked to write "a single SQL query" to produce some output. Ensure that your query uses the *minimum* number of tables possible, and that it produces *exactly* the output specified (no extra columns). A single SQL query may have more than one `SELECT` statement only if it uses a subquery and/or a compound-select statement.

Use only SQLite (not other variants of SQL) in your answer, and do not incorporate any features that are not required by the question. You may use table aliases and column aliases if they are necessary or improve the readability of the query, but otherwise only rename columns if the question asks for them to be renamed.

Be as precise as possible with punctuation such as commas, quotes and terminating semi-colons, because these play an important role in determining the meaning of a query.

(1) Routine questions on `SELECT`, `WHERE` and `ORDER`

Write a single SQL query that will list the *title* and *budget* for every project with a budget of at least one million dollars, listing them in decreasing order of budget.

```sql
SELECT title, budget
FROM Project
WHERE budget >= 1000000
ORDER BY budget DESC;
```

Write a single SQL query that will list the *charityName* and *type* of each charity that is dedicated to either medical or animal issues, listing them in alphabetical order of charity name.

```sql
SELECT charityName, type
FROM Charity
WHERE type == 'medical' OR type == 'animal'
ORDER BY charityName ASC;
```

---

(2) Routine question involving multiple-table `JOIN` and `SELECT DISTINCT`

Write a single SQL query that lists the *names*, once each, of all the donors who have donated to a medical or animal charity.

```
SELECT DISTINCT donorName
FROM Donor JOIN gives USING (donorId)
JOIN Charity USING (charityName)
WHERE type == 'medical' OR type == 'animal';
```

Write a single SQL query that lists the *titles*, once each, of all the projects that have been funded by a welfare or research charity.

```
SELECT DISTINCT title
FROM Project JOIN funds USING (projectId)
JOIN Charity USING (charityName)
WHERE type == 'research' OR type == 'welfare';
```

(3) Basic GROUP BY and aggregate functions

Write a single SQL query that lists, for each country, the *country* and the *total amount* of money donated by donors from that country. The output table should have columns named `country` and `totalAmount`.

```
SELECT country, SUM(amount) AS totalAmount
FROM Donor JOIN gives USING (donorId)
GROUP BY country;
```

Write a single SQL query that lists, for each project, the *title* and the *total amount* of funding obtained for that project. The output table should have columns named `title` and `amountFunded`.

```
SELECT title, SUM(amount) AS amountFunded
FROM Project JOIN funds USING (projectId)
GROUP BY projectId;
```

---

(4) GROUP BY and HAVING

Write a single SQL query that lists, for each combination of calendar year and charity, the *year* (in 4-digit YYYY form), the *charity name* and the *total value* of all donations to that charity in that calendar year, provided this total value is more than $10 million dollars. The rows should be sorted so that the most recent years are first, and within each year, the charities with the highest total donations first.

```sql
SELECT substr(date,1,4) AS yr, charityName, SUM(amount) as total
FROM gives
GROUP BY yr, charityName
HAVING total > 10000000
ORDER BY yr DESC, total DESC;
```

(5) Finding the maximum in a list

Write a single SQL query that lists, for each project, its *title* and the *charityName* of the charity that has provided the *most funding*. If there are multiple charities that have funded equal-maximum amounts, then there should be one row for each of them.

```
SELECT title, charityName
FROM Project JOIN funds USING (projectId)
WHERE (projectId, amount) IN
(SELECT projectId, MAX(amount)
FROM funds
GROUP BY projectId);
```

Write a single SQL query that lists, for each charity, its *charityName* and the *donorName* of the donor who has made the biggest *single donation* to that charity. If there are multiple donors that have made equal-maximum single donations, then there should be one output row for each of them.

```
SELECT charityName, donorName
FROM Donor JOIN gives USING (donorId)
WHERE (charityName, amount) IN
(SELECT charityName, MAX(amount)
FROM gives
GROUP BY charityName);
```

(6) Using an `OUTER JOIN`

Write a single SQL query that lists, for each project, its *title* and the *number of charities* that have contributed funding to the project. Make sure to include projects that have not been allocated any funding.

```
SELECT title, COUNT(amount)
FROM Project LEFT JOIN funds USING (projectId)
GROUP BY projectId, title;
```

Write a single SQL query that lists, for each charity, its *name* and the *number of donors* that have donated to this charity. A donor who has made multiple donations to the same charity should only be counted once. Make sure to include charities that have not received any donations.

```
SELECT charityName, COUNT(DISTINCT donorId)
FROM Charity LEFT JOIN gives USING (charityName),
GROUP BY charityName;
```

(7) Subqueries easy

Using an uncorrelated subquery, write a single SQL query that lists the *names*, once each, of the donors that have donated *only* to medical charities.

```
SELECT DISTINCT donorName
FROM Donor
WHERE donorId NOT IN
(SELECT donorId FROM gives JOIN Charity USING (charityName)
WHERE type IN ('welfare','animal','research'));
```

---

(8) Using `CASE`

High-value donations are classified into *donation tiers* as follows: a donation of more than $100000 is a *bronze* donation, one of more than $250000 is a *silver* donation and one of more than $1 million dollars is a *gold* donation.

Write a single SQL query that lists, for each donation, the *name* of the donor, the *name* of the charity concerned, and the *donation tier* of that particular donation, which should be one of the strings `'bronze'`, `'silver'` or `'gold'` for high-value donations and `NULL` otherwise.

```
SELECT
donorName, charityName,
CASE
WHEN amount >= 1000000 THEN 'gold'
WHEN amount >= 250000 THEN 'silver'
WHEN amount >= 100000 THEN 'bronze'
ELSE NULL
END
FROM Donor JOIN gives USING (donorId);
```

Projects classify their funding charities into three funding tiers, depending on what proportion of the budget has been funded by that charity. A charity that funds more than 20% of a project's budget is called a *supporter*, a charity that funds more than 50% of a project's budget is called a *sponsor* and a charity that funds more than 80% of a project's budget is called a *patron*.

Write a single SQL query that lists, for each project and charity that has funded that project, the *title* of the project, the *name of the charity* and the *funding tier*, which should be one of the strings `supporter`, `sponsor` or `patron` if the amount of funding is sufficiently high and `NULL` otherwise.

```
SELECT
title,
CASE
WHEN (amount > 0.8*budget) THEN 'patron'
WHEN (amount > 0.5*budget) THEN 'sponsor'
WHEN (amount > 0.2*budget) THEN 'supporter'
ELSE NULL
END
FROM Project JOIN funds USING (projectId);
```

---

(9) *Without using a subquery*, write a single SQL query that lists, for each donor, the donor's *name*, their *medical donations* (i.e., the total value of their donations to medical charities) and their *nonmedical donations* (i.e., the total value of their donations to the other 3 types of charity).

```
SELECT donorName,
     SUM(IIF(type == 'medical',amount,0)),
     SUM(IIF(type != 'medical',amount,0))
     FROM Donor JOIN gives USING (donorId)
JOIN Charity USING (charityName)
GROUP BY donorId;
```

(10) Difficult subquery question
Write a single SQL query that lists the *projectId* of the projects that have received funding from *every* charity.

```
# Counting charities and funding charities
SELECT projectId
FROM Projects P
WHERE
(SELECT COUNT(*) FROM Charity)
=
(SELECT COUNT(*) FROM funds WHERE funds.projectId = P.projectId);
```

```
# Non-existence of non-funding charity
SELECT P.projectId
FROM Projects P
WHERE NOT EXISTS
(SELECT * FROM Charity
WHERE charityName NOT IN (SELECT charityName
           FROM funds WHERE funds.projectId = P.projectId);
```

(11) DDL and data integrity

Write the DDL statements necessary to create the tables `Charity` and `gives` including suitable data integrity features as follows. Each Charity should have a valid type (one of the four types listed), each donation must be at least $1000 and must refer to a valid donor and charity. Charities sometimes change their name, and if this happens rows referring to the charity must be changed in the same way. Charities sometimes cease operating in which case they are deleted from the database, and all entries referring to that charity should be replaced with `NULL`.

```sql
CREATE TABLE Charity (
 charityName TEXT PRIMARY KEY,
 type TEXT,
 CHECK (type IN ('medical','animal','welfare','research')),
);

CREATE TABLE gives (
  donorId INTEGER,
  charityName TEXT,
  amount INTEGER,
  date TEXT,
  CHECK (amount >=1000),
  FOREIGN KEY (donorId) REFERENCES Donor (donorId),
  FOREIGN KEY (charityName) REFERENCES Charity (charityName)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);
```