Customer Support Data Labeling System

Goal: Build a relational database to store customer support tickets, flagged sensitive information, and labeling decisions.

Tables:

- `tickets(ticket_id, customer_id, issue_text, created_at)`
- `labels(ticket_id, label, reviewer_id, flagged_sensitive BOOLEAN)`
- `reviewers(reviewer_id, name, role)`

Queries:

- Find the percentage of tickets flagged as sensitive.
- Identify reviewers with the highest correction rate.

a. Create database

```
CREATE DATABASE customer_support;
USE customer_support;
```

```
mysql> CREATE DATABASE customer_support;
Query OK, 1 row affected (0.10 sec)

mysql> USE customer_support;
Database changed
```

b. Create tables

```
-- Table for customer support tickets
CREATE TABLE tickets (
    ticket_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    issue_text TEXT,
    created_at DATETIME
);

-- Table for reviewers
CREATE TABLE reviewers (
    reviewer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    role VARCHAR(50)
);

-- Table for labels applied to tickets
CREATE TABLE labels (
    label_id INT AUTO_INCREMENT PRIMARY KEY,
    ticket_id INT,
    reviewer_id INT,
    label VARCHAR(50),
    flagged_sensitive BOOLEAN,
    reviewed_at DATETIME,
    FOREIGN KEY (ticket_id) REFERENCES tickets(ticket_id),
    FOREIGN KEY (reviewer_id) REFERENCES reviewers(reviewer_id)
);
```

```
mysql> CREATE TABLE tickets ( ticket_id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT,
 issue_text TEXT, created_at DATETIME );
Query OK, 0 rows affected (0.21 sec)

mysql> CREATE TABLE reviewers ( reviewer_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR
(100), role VARCHAR(50) );
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE labels ( label_id INT AUTO_INCREMENT PRIMARY KEY, ticket_id INT, rev
iewer_id INT, label VARCHAR(50), flagged_sensitive BOOLEAN, reviewed_at DATETIME, FOREIG
N KEY (ticket_id) REFERENCES tickets(ticket_id), FOREIGN KEY (reviewer_id) REFERENCES re
viewers(reviewer_id) );
Query OK, 0 rows affected (0.13 sec)
```

c. Insert sample data

```
-- Table for customer support tickets
CREATE TABLE tickets (
    ticket_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    issue_text TEXT,
    created_at DATETIME
);

-- Table for reviewers
CREATE TABLE reviewers (
    reviewer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    role VARCHAR(50)
);

-- Table for labels applied to tickets
CREATE TABLE labels (
    label_id INT AUTO_INCREMENT PRIMARY KEY,
    ticket_id INT,
    reviewer_id INT,
    label VARCHAR(50),
    flagged_sensitive BOOLEAN,
    reviewed_at DATETIME,
    FOREIGN KEY (ticket_id) REFERENCES tickets(ticket_id),
    FOREIGN KEY (reviewer_id) REFERENCES reviewers(reviewer_id)
);
```

```
mysql> INSERT INTO tickets (customer_id, issue_text, created_at) VALUES (101, 'Password
reset request', '2025-10-01 09:15:00'), (102, 'Billing issue with credit card', '2025-10
-02 14:30:00'), (103, 'Complaint about service quality', '2025-10-03 11:45:00');
Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO reviewers (name, role) VALUES ('Alice', 'Data Labeler'), ('Bob', 'Rev
iewer');
Query OK, 2 rows affected (0.03 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO labels (ticket_id, reviewer_id, label, flagged_sensitive, reviewed_at
)
    -> VALUES
    -> (1, 1, 'Technical Support', FALSE, '2025-10-01 10:00:00'),
    -> (2, 2, 'Billing', TRUE, '2025-10-02 15:00:00'),
    -> (3, 1, 'Complaint', FALSE, '2025-10-03 12:00:00');
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

d. Example queries

```
-- Table for customer support tickets
CREATE TABLE tickets (
```

```
    ticket_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    issue_text TEXT,
    created_at DATETIME
);

-- Table for reviewers
CREATE TABLE reviewers (
    reviewer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    role VARCHAR(50)
);

-- Table for labels applied to tickets
CREATE TABLE labels (
    label_id INT AUTO_INCREMENT PRIMARY KEY,
    ticket_id INT,
    reviewer_id INT,
    label VARCHAR(50),
    flagged_sensitive BOOLEAN,
    reviewed_at DATETIME,
    FOREIGN KEY (ticket_id) REFERENCES tickets(ticket_id),
    FOREIGN KEY (reviewer_id) REFERENCES reviewers(reviewer_id)
);
```

```
mysql> SELECT (SUM(flagged_sensitive)/COUNT(*))*100 AS sensitive_percentage FROM labels;
+----------------------+
| sensitive_percentage |
+----------------------+
|              33.3333 |
+----------------------+
1 row in set (0.04 sec)

mysql> SELECT r.name, COUNT(l.flagged_sensitive) AS corrections FROM reviewers r JOIN labels l ON r.reviewer_id = l.reviewer_id WHERE l.flagged_
sensitive = TRUE GROUP BY r.name ORDER BY corrections DESC;
+------+-------------+
| name | corrections |
+------+-------------+
| Bob  |           1 |
+------+-------------+
1 row in set (0.04 sec)

mysql> SELECT t.ticket_id, t.issue_text, l.label, l.flagged_sensitive, r.name AS reviewer FROM tickets t JOIN labels l ON t.ticket_id = l.ticket
_id JOIN reviewers r ON l.reviewer_id = r.reviewer_id;
+-----------+---------------------------------+-------------------+-------------------+----------+
| ticket_id | issue_text                      | label             | flagged_sensitive | reviewer |
+-----------+---------------------------------+-------------------+-------------------+----------+
|         1 | Password reset request          | Technical Support |                 0 | Alice    |
|         3 | Complaint about service quality | Complaint         |                 0 | Alice    |
|         2 | Billing issue with credit card  | Billing           |                 1 | Bob      |
+-----------+---------------------------------+-------------------+-------------------+----------+
3 rows in set (0.00 sec)
```