Assignment

MODULE: 8

Local Storage and Persistence

Q.1 Explain the difference between local storage options (shared_preferences, SQLite, Hive).

➢ **Ans.**

1. shared_preferences

Best for: Small key-value pairs (like user settings or preferences)

Advantage :

- Lightweight and easy to use.

- Stores simple data types: `int`, `double`, `bool`, `String`, and `List<String>`.

- Persistent across sessions.

- Uses native platform storage (NSUserDefaults on iOS, SharedPreferences on Android).

Disadvantage :

- Not suitable for large or complex datasets.

- No query support or data relationships.

Example use case:

Remembering if the user is logged in.

Storing theme settings (light/dark mode).

---

2. SQLite

Best for: Structured data with relationships, complex queries

✓ Advantage:

- Full relational database.
- Can perform complex queries with SQL.
- Great for large datasets and structured records.
- More boilerplate and setup.

Example use case:

- Offline storage of structured data like contacts, tasks, or inventory.
- Apps needing sorting/filtering/searching large data.

****

Q.2 Describe CRUD operations and how they are implemented in SQLite or Hive ?

➢ **Ans.**

CRUD stands for the four basic operations you can perform on persistent data:

Operation	Meaning	Purpose
C	Create	Add new data
R	Read	Retrieve or fetch data
U	Update	Modify existing data
D	Delete	Remove data

How CRUD Is Implemented in SQLite

SQLite is a **relational database**, so CRUD is performed using **SQL queries**.

✓ 1. Create

```
await db.insert(
  'users',
  {'id': 1, 'name': 'Alice'},
);
```

✓ 2. Read

```
List<Map<String, dynamic>> users = await db.query('users');
```

✓ 3. Update

```
await db.update(
  'users',
  {'name': 'Bob'},
  where: 'id = ?',
  whereArgs: [1],
);
```

✓ 4. Delete

```
await db.delete(
  'users',
  where: 'id = ?',
  whereArgs: [1],
);
```

Q.3 Explain the advantages and use cases for shared_preferences ?

➢ **Ans.**

✓ Advantages of **`shared_preference`** :

Simple to use – Easy API for storing key-value pairs.

Lightweight – Minimal setup and fast performance.

Persistent – Data remains after app restarts.

Cross-platform – Works on Android, iOS, and more.

No native database needed – Uses platform-specific storage internally.

Use Cases:

- Saving **user login status** (e.g. `isLoggedIn = true`)

- Storing **app theme or language preference**

- Remembering **onboarding screen status**

- Keeping **small user settings** or toggles