Q1: Statements 1, 2, and 3 Are Correct

Analysis of Each Statement

1. **✓ Correct** –
   Referenced in `docker-entrypoint.sh` (line 10):
   ```
   1. gcsfuse --key-file=$GOOGLE_APPLICATION_CREDENTIALS $GCS_BUCKET_NAME
      /mnt/gcs_data
   ```
2. **✓ Correct** –
   The command `mount --bind /mnt/gcs_data/images /app/cheese_dataset` maps the local view of remote data into the application workspace.
   Specifically, in `docker-entrypoint.sh` (line 13):
   ```
   1. mount --bind /mnt/gcs_data/images /app/cheese_dataset
   ```
3. **✓ Correct** –
   DVC recognizes `/app/cheese_dataset` as a local path, but all reads and writes are routed through the FUSE-mounted GCS directory.
4. **✗ Incorrect** –
   The bucket is not mounted directly into `/app`. It is mounted first at `/mnt/gcs_data` and then bind-mounted.
   Furthermore, data in `/app` **does** persist on the host due to the following volume mount:
   ```
   1. -v "$BASE_DIR":/app
   ```
5. **✗ Incorrect** –
   Secrets are volume-mounted via:
   ```
   1. -v "$SECRETS_DIR":/secrets
   ```

   (see `docker-shell.sh`, line 22), rather than being embedded in the image.

Q2: Correct Answer — H, I, and F Can Be Dropped

Analysis

Lines That Can Be Dropped:

- **H)** `-e GCP_PROJECT` – Not used in `docker-entrypoint.sh` or any visible code.
- **I)** `-e GCP_ZONE` – Not used in `docker-entrypoint.sh` or any visible code.
- **F)** `-v ~/.gitconfig:/etc/gitconfig` – Can be omitted if all Git operations are performed outside the container.

Lines That Cannot Be Dropped:

- **A, B, C** – Required for FUSE mounting (`docker-entrypoint.sh`: lines 10, 13).
- **D** – Mounts source code and DVC files.
- **E** – Mounts service account credentials (used in `docker-entrypoint.sh`: lines 8, 10).
- **G** – Used in `docker-entrypoint.sh`: lines 8, 10.
- **J** – Used in `docker-entrypoint.sh`: line 10.
- **K** – The image name.

Answer: H, I, and F.

Q3: Correct Answer — B

Explanation

The mount occurs in `docker-entrypoint.sh` (lines 10–13):

```
gcsfuse --key-file=$GOOGLE_APPLICATION_CREDENTIALS $GCS_BUCKET_NAME /mnt/gcs_data
mount --bind /mnt/gcs_data/dvc_store/images /app/cheese_dataset
```

First, `gcsfuse` mounts the GCS bucket to `/mnt/gcs_data`, and then `mount --bind` maps the images subdirectory to `/app/cheese_dataset`.

Q4: No, It Does Not Appear in GCS

Explanation

When adding a file **outside the container** to the `cheese_dataset` folder, it is stored on the host filesystem.
However, **inside the container**, `/app/cheese_dataset` is overlaid by the bind mount defined in `docker-entrypoint.sh` (line 13):

```
mount --bind /mnt/gcs_data/dvc_store/images /app/cheese_dataset
```

This bind mount hides the host directory's contents (including any new files) and instead exposes the data from the GCS bucket.

Responsible Lines:

- `docker-entrypoint.sh:13` – Bind mount overlays the local directory.
- `docker-shell.sh:21` – Volume mount maps the host directory:
- `-v "$BASE_DIR":/app`

As a result, the file exists on the host but is not visible within the container or in GCS due to the overlay behavior.

Q5: Yes, It Is Necessary

Explanation

Line 12 of `docker-entrypoint.sh` creates the required mount point:

```
mkdir -p /app/cheese_dataset
```

The subsequent bind mount (line 13) requires this directory to exist.
If this line were removed and `/app/cheese_dataset` did not already exist, the `mount --bind` command would fail with an error such as:

```
mount point does not exist
```

Thus, `mkdir -p` ensures the mount point exists regardless of the host directory's state.

Q6: No, It Does Not Need to Be Performed Outside the Container

Explanation

While the README (line 119) recommends running Git commands outside the container, technically they can be executed inside because:

1. Git configuration is mounted:
2. `-v ~/.gitconfig:/etc/gitconfig`

   (see `docker-shell.sh:23`)

3. The repository is volume-mounted:
4. `-v "$BASE_DIR":/app`

   (see `docker-shell.sh:21`)

5. Consequently, Git operations inside the container affect the same repository as on the host.

However, these operations are **not automated** by the code.
The user must manually execute:

```
git add .
git commit -m "dataset updates..."
git tag -a "dataset_v20" -m "tag dataset"
git push --atomic origin main dataset_v20
```

The infrastructure (mounted Git config and repo) is provided, but the README suggests running these commands outside the container for simplicity and to avoid potential permission issues—not due to technical necessity.