```
password=\"password\",
host=\"localhost\",
port=\"5432\")
cur = conn.cursor()
cur.execute(\"\"\"
CREATE TABLE instagram_posts (id SERIAL PRIMARY KEY,        post_id INTEGER REFERENCES
    posts(id),        engagement_metrics JSONB);
\"\"\")\n\nconn.commit()
cur.close()
conn.close()
```

Assistant –> Code Review
Your approach is good! Consider adding `NOT NULL`, timestamps, and indexes for performance . . .

```
CREATE TABLE IF NOT EXISTS instagram_posts (
  id SERIAL PRIMARY KEY,
  post_id INTEGER NOT NULL REFERENCES posts(id),
  engagement_metrics JSONB,
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
CREATE INDEX IF NOT EXISTS idx_instagram_posts_post_id
  ON instagram_posts(post_id);
```

[Trunicated ...] plus safe migration with INSERT SELECT to backfill existing posts.

User –> Deployment
I'm trying to deploy my updated posting services to an AWS EC2 t3.medium instance with 2 vCPUs and 4GB RAM, but I'm running into some issues with the deployment process. Can you help me figure out how to properly deploy my application to this instance, considering the DevOps and deployment aspects? I've been using Docker to containerize my application, and I'm not sure what changes I need to make to ensure a smooth deployment. Here's a snippet of my current Dockerfile:

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

Assistant –> Deployment Guidance
Install Docker on EC2, push image to registry, pull with pinned tag, and run with `--restart=always` [Trunicated ...] add healthchecks and a minimal `systemd` unit [Trunicated ...]

User (Follow-up)
hmm, what's the best way to automate the pulling and running of the Docker container?

Assistant –> CI/CD Automation
Use GitHub Actions with SSH deploy step [Trunicated ...]

```
name: Deploy
on: { push: { branches: [main] } }
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: |
          ssh -i id_rsa ec2-user@EC2_DNS \
          "docker stop socauto || true && docker rm socauto || true &&
           docker pull username/your-app:sha-${GITHUB_SHA} &&
           docker run -d --restart=always -p 80:80 username/your-app:sha-${GITHUB_SHA}"
```

User (Follow-up)
hmm, what's the best way to handle environment variables for different stages like dev and prod?