Diaz – Documentation Technical Test

Table of Contents

Introduction

This project is a submission for a technical test that demonstrates skills in backend development, automation scripting, and data processing. It includes three core tasks:

1. Building a simple backend server using Node.js/Express to manage data via a REST API.
2. Creating automation scripts using a cron job for periodic data collection and cleansing.
3. Writing a series of SQL queries to manipulate and retrieve data from a table.

Environment Setup

To run the project, please ensure that the following dependencies are installed in your environment. You can use Node.js & npm, a Linux-based environment

Requirements

- Node.js and npm.
- Postman.
- Dependencies (npm install).
- Linux based environment.
- SQL Client (MySQL Workbench).
- Database Server : MySQL Server or other compatible database.

Point 1: Backend Development

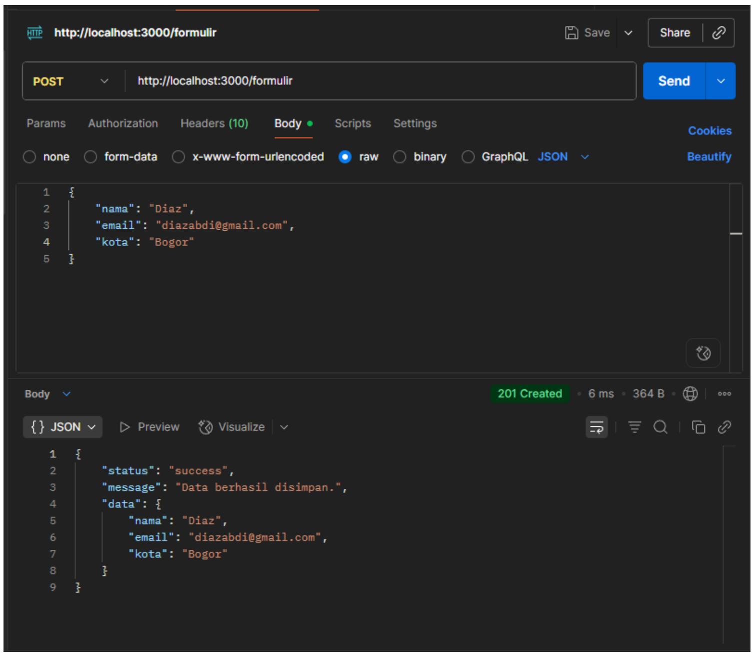
Description

The first task is a small backend server built with Node.js and Express. It's designed to handle form submissions. The script receives data from a form, saves it, and can send back all the data it has stored.

How to Run

1. Run `npm install` to install all the necessary libraries and packages listed in the package.json file, such as Express

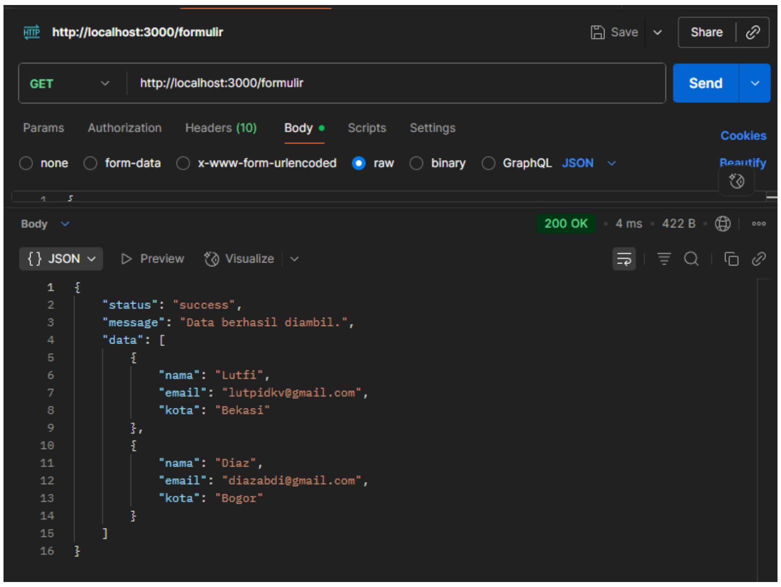
2. Run the server using `node server.js` , if the server starts successfully, you will see a confirmation message in your terminal: `Server berjalan di http://localhost:3000`
3. Testing the API endpoint using a tool like Postman
   - POST /formulir : To submit new form data.



```
{
    "nama": "Diaz",
    "email": "diazabdi@gmail.com",
    "kota": "Bogor"
}
```

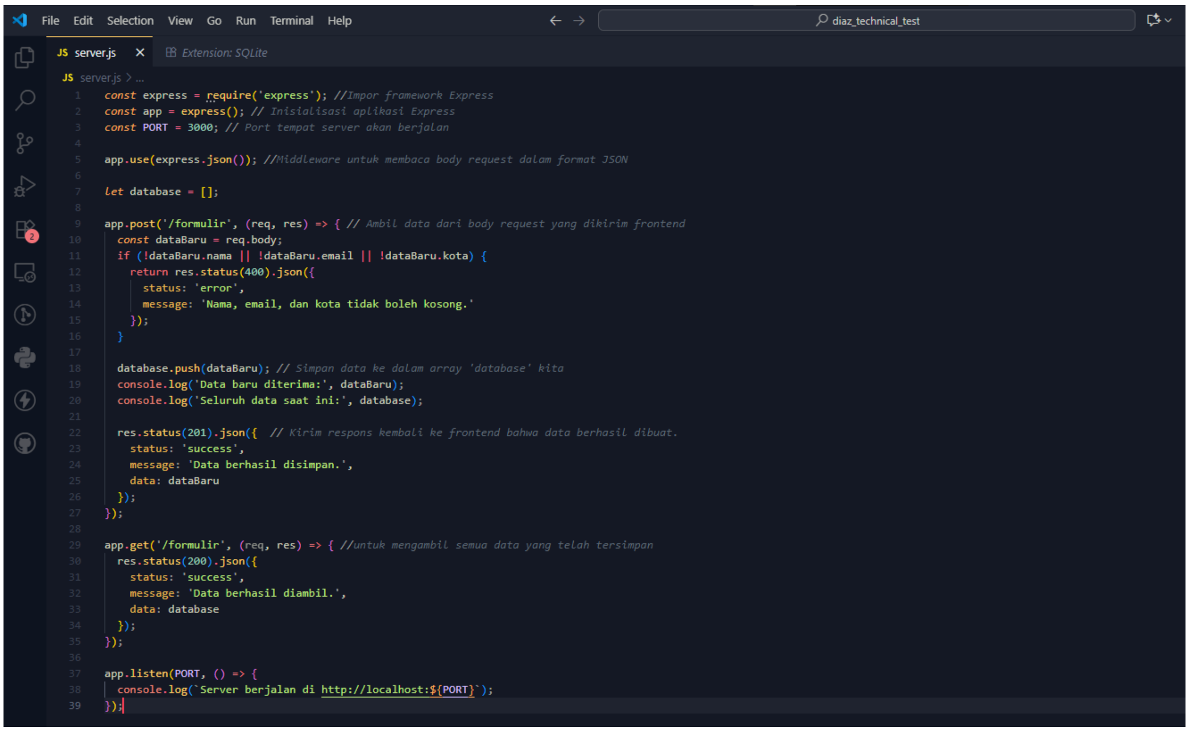
```
{
    "status": "success",
    "message": "Data berhasil disimpan.",
    "data": {
        "nama": "Diaz",
        "email": "diazabdi@gmail.com",
        "kota": "Bogor"
    }
}
```

   - GET /formulir : To retrieve all submitted data.



```
{
    "status": "success",
    "message": "Data berhasil diambil.",
    "data": [
        {
            "nama": "Lutfi",
            "email": "lutpidkv@gmail.com",
            "kota": "Bekasi"
        },
        {
            "nama": "Diaz",
            "email": "diazabdi@gmail.com",
            "kota": "Bogor"
        }
    ]
}
```

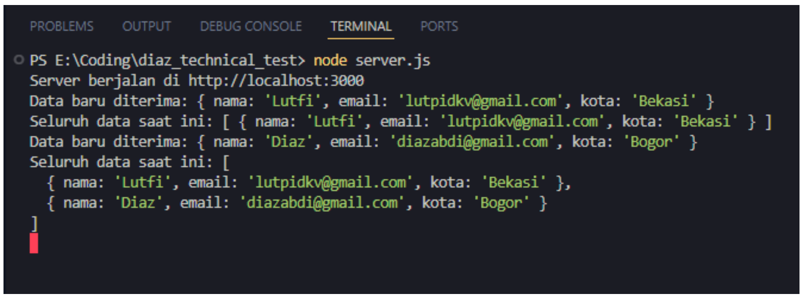
Code Explanation



```javascript
const express = require('express'); //Impor framework Express
const app = express(); // Inisialisasi aplikasi Express
const PORT = 3000; // Port tempat server akan berjalan

app.use(express.json()); //Middleware untuk membaca body request dalam format JSON

let database = [];

app.post('/formulir', (req, res) => { // Ambil data dari body request yang dikirim frontend
  const dataBaru = req.body;
  if (!dataBaru.nama || !dataBaru.email || !dataBaru.kota) {
    return res.status(400).json({
      status: 'error',
      message: 'Nama, email, dan kota tidak boleh kosong.'
    });
  }

  database.push(dataBaru); // Simpan data ke dalam array 'database' kita
  console.log('Data baru diterima:', dataBaru);
  console.log('Seluruh data saat ini:', database);

  res.status(201).json({  // Kirim respons kembali ke frontend bahwa data berhasil dibuat.
    status: 'success',
    message: 'Data berhasil disimpan.',
    data: dataBaru
  });
});

app.get('/formulir', (req, res) => { //untuk mengambil semua data yang telah tersimpan
  res.status(200).json({
    status: 'success',
    message: 'Data berhasil diambil.',
    data: database
  });
});

app.listen(PORT, () => {
  console.log(`Server berjalan di http://localhost:${PORT}`);
});
```

- The app.post endpoint is for receiving data from the frontend's request body and storing it in a simple array.
- The app.get endpoint retrieves all the stored data and sends it back as a response.

Output



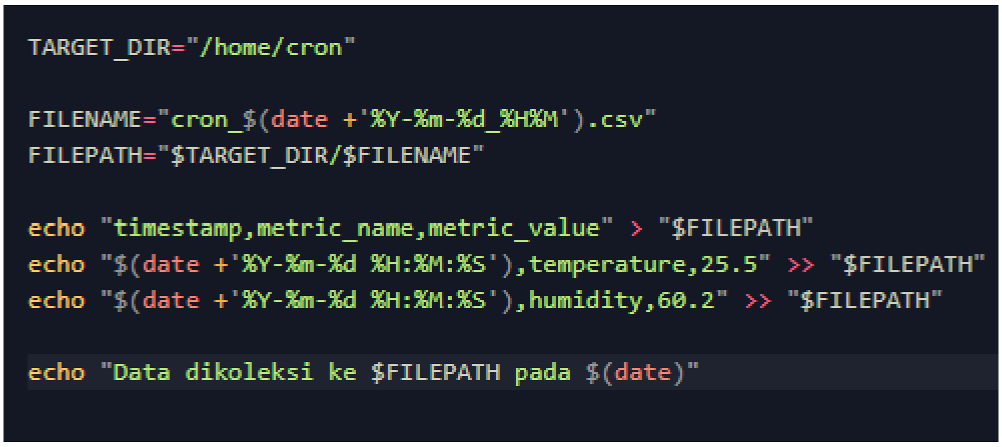
```
PS E:\Coding\diaz_technical_test> node server.js
Server berjalan di http://localhost:3000
Data baru diterima: { nama: 'Lutfi', email: 'lutpidkv@gmail.com', kota: 'Bekasi' }
Seluruh data saat ini: [ { nama: 'Lutfi', email: 'lutpidkv@gmail.com', kota: 'Bekasi' } ]
Data baru diterima: { nama: 'Diaz', email: 'diazabdi@gmail.com', kota: 'Bogor' }
Seluruh data saat ini: [
  { nama: 'Lutfi', email: 'lutpidkv@gmail.com', kota: 'Bekasi' },
  { nama: 'Diaz', email: 'diazabdi@gmail.com', kota: 'Bogor' }
]
```

Point 2: Automation Testing

Description

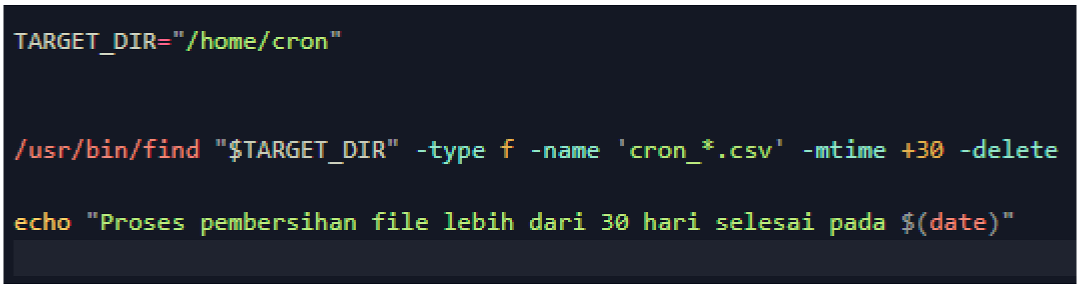
This project contains two shell scripts to automate the process of periodic data collection and the automatic cleanup of old data using cron in a Linus environment.

- **collect_data.sh**: A script to collect data and save it to a new .csv file in /home/cron. The filename is dynamically generated based on the execution date and time.



```
TARGET_DIR="/home/cron"

FILENAME="cron_$(date +'%Y-%m-%d_%H%M').csv"
FILEPATH="$TARGET_DIR/$FILENAME"

echo "timestamp,metric_name,metric_value" > "$FILEPATH"
echo "$(date +'%Y-%m-%d %H:%M:%S'),temperature,25.5" >> "$FILEPATH"
echo "$(date +'%Y-%m-%d %H:%M:%S'),humidity,60.2" >> "$FILEPATH"

echo "Data dikoleksi ke $FILEPATH pada $(date)"
```

- **cleanup_data.sh**: A script to check the /home/cron directory and automatically delete .csv files older than 30 days.



```
TARGET_DIR="/home/cron"


/usr/bin/find "$TARGET_DIR" -type f -name 'cron_*.csv' -mtime +30 -delete

echo "Proses pembersihan file lebih dari 30 hari selesai pada $(date)"
```

Configuration

1. Copy the scripts to your home directory(~/)
2. Create the directory for storing the CSV data files and set the correct permissions
   ```
   sudo mkdir -p /home/cron
   sudo chown $(whoami):$(whoami) /home/cron
   ```
3. Make the scripts executable
   ```
   chmod +x ~/collect_data.sh
   chmod +x ~/cleanup_data.sh
   ```
4. Set up the cron schedule
   - Open crontab editor: `crontab -e`
   - Add the following lines at the end of the file, and replace YOUR_USERNAME with your actual username.

     ```
     # Run the data collection script 3 times a day (08:00, 12:00, 15:00)
     0 8 * * * /home/YOUR_USERNAME/collect_data.sh
     0 12 * * * /home/YOUR_USERNAME/collect_data.sh
     0 15 * * * /home/YOUR_USERNAME/collect_data.sh

     # Run the cleanup script every day at 03:05 AM
     5 3 * * * /home/YOUR_USERNAME/cleanup_data.sh
     ```

   - Save it and close the editor

How to Run

No manual run command. Cron will automatically run the scripts according to schedule

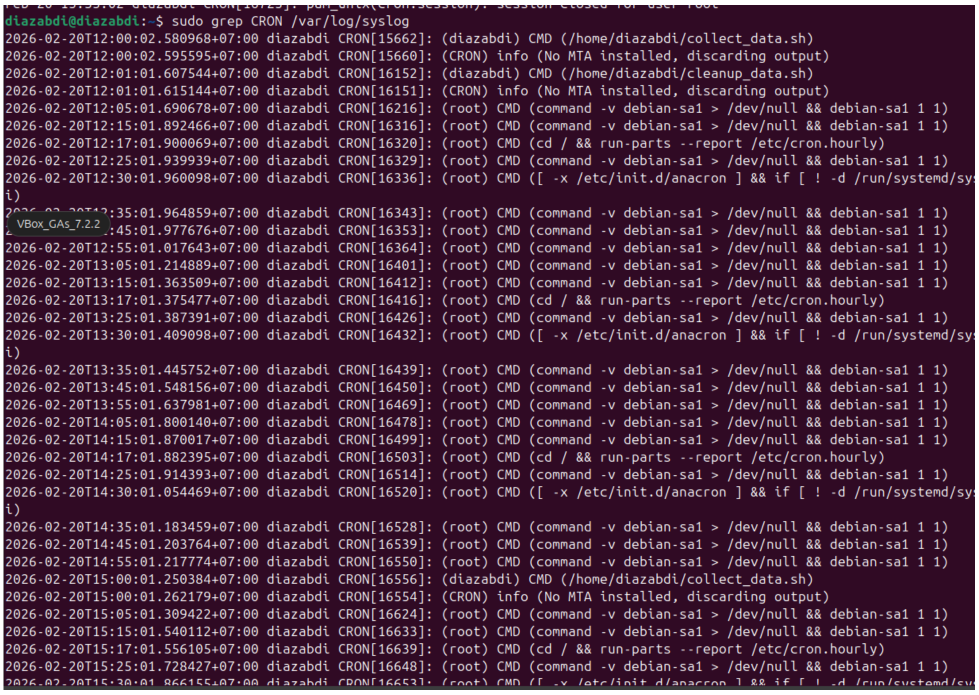
- **Collect_data.sh** will run at 8 AM, 12 PM, and 3 PM every day.
- **Cleanup_data.sh** will run at 03:05 AM everyday (if the computer is on).

Verification

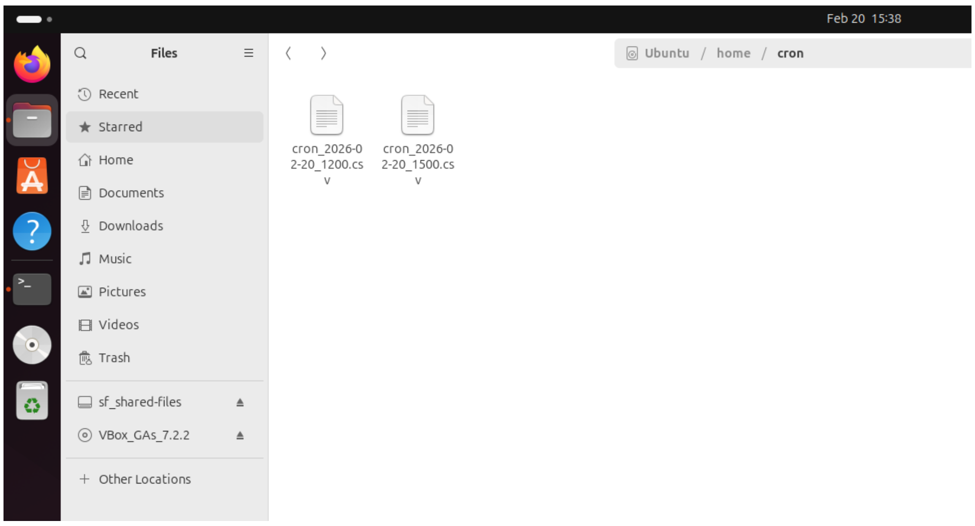
To verify that the system is working correctly:

- Check the contents of the /home/cron directory after an execution time. A new CSV file should appear.
  `ls -lh /home/cron`
- View the system log to confirm that cron has executed your commands
  `sudo grep CRON /var/log/syslog`

Output



```
Feb 20 15:35:02 diazabdi CRON[16723]: pam_unix(cron:session): session closed for user root
diazabdi@diazabdi:~$ sudo grep CRON /var/log/syslog
2026-02-20T12:00:02.580968+07:00 diazabdi CRON[15662]: (diazabdi) CMD (/home/diazabdi/collect_data.sh)
2026-02-20T12:00:02.595595+07:00 diazabdi CRON[15660]: (CRON) info (No MTA installed, discarding output)
2026-02-20T12:01:01.607544+07:00 diazabdi CRON[16152]: (diazabdi) CMD (/home/diazabdi/cleanup_data.sh)
2026-02-20T12:01:01.615144+07:00 diazabdi CRON[16151]: (CRON) info (No MTA installed, discarding output)
2026-02-20T12:05:01.690678+07:00 diazabdi CRON[16216]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T12:15:01.892466+07:00 diazabdi CRON[16316]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T12:17:01.900069+07:00 diazabdi CRON[16320]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)
2026-02-20T12:25:01.939939+07:00 diazabdi CRON[16329]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T12:30:01.960098+07:00 diazabdi CRON[16336]: (root) CMD ([ -x /etc/init.d/anacron ] && if [ ! -d /run/systemd/sy
i)
2026-02-20T12:35:01.964859+07:00 diazabdi CRON[16343]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
VBox_GAs_7.2.2 :45:01.977676+07:00 diazabdi CRON[16353]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T12:55:01.017643+07:00 diazabdi CRON[16364]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T13:05:01.214889+07:00 diazabdi CRON[16401]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T13:15:01.363509+07:00 diazabdi CRON[16412]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T13:17:01.375477+07:00 diazabdi CRON[16416]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)
2026-02-20T13:25:01.387391+07:00 diazabdi CRON[16426]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T13:30:01.409098+07:00 diazabdi CRON[16432]: (root) CMD ([ -x /etc/init.d/anacron ] && if [ ! -d /run/systemd/sy
i)
2026-02-20T13:35:01.445752+07:00 diazabdi CRON[16439]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T13:45:01.548156+07:00 diazabdi CRON[16450]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T13:55:01.637981+07:00 diazabdi CRON[16469]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T14:05:01.800140+07:00 diazabdi CRON[16478]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T14:15:01.870017+07:00 diazabdi CRON[16499]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T14:17:01.882395+07:00 diazabdi CRON[16503]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)
2026-02-20T14:25:01.914393+07:00 diazabdi CRON[16514]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T14:30:01.054469+07:00 diazabdi CRON[16520]: (root) CMD ([ -x /etc/init.d/anacron ] && if [ ! -d /run/systemd/sy
i)
2026-02-20T14:35:01.183459+07:00 diazabdi CRON[16528]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T14:45:01.203764+07:00 diazabdi CRON[16539]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T14:55:01.217774+07:00 diazabdi CRON[16550]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T15:00:01.250384+07:00 diazabdi CRON[16556]: (diazabdi) CMD (/home/diazabdi/collect_data.sh)
2026-02-20T15:00:01.262179+07:00 diazabdi CRON[16554]: (CRON) info (No MTA installed, discarding output)
2026-02-20T15:05:01.309422+07:00 diazabdi CRON[16624]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T15:15:01.540112+07:00 diazabdi CRON[16633]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T15:17:01.556105+07:00 diazabdi CRON[16639]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)
2026-02-20T15:25:01.728427+07:00 diazabdi CRON[16648]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
2026-02-20T15:30:01.866155+07:00 diazabdi CRON[16653]: (root) CMD ([ -x /etc/init.d/anacron ] && if [ ! -d /run/systemd/sy
```



cron_2026-02-20_1200.csv
cron_2026-02-20_1500.csv

Point 3: Data Processing

Description

This project contains a set of SQL Scripts to create, manipulate, and query a simple employee data table. It serves CREATE, INSERT, UPDATE, SELECT operations.

Configuration

The SQL_Script.sql is designed to handle most of the configuration automatically. The only manual setup step is to ensure your database server is running.

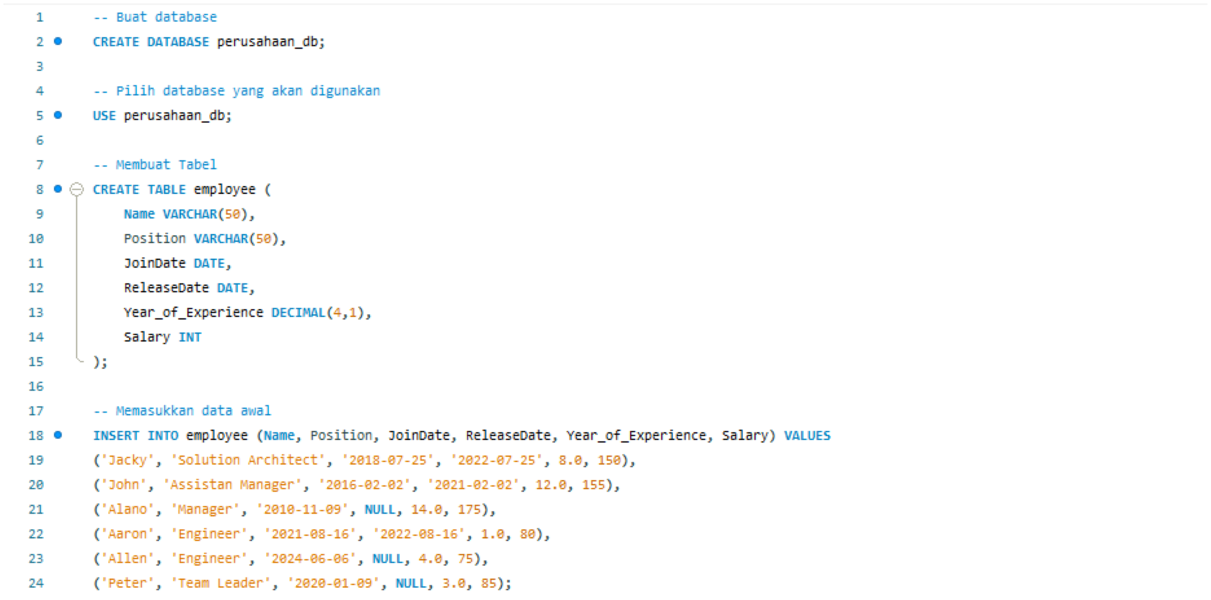
The script will automatically perform the following actions:

1. Create a new database
2. Switch to use the database
3. Create the employee table with the correct schema

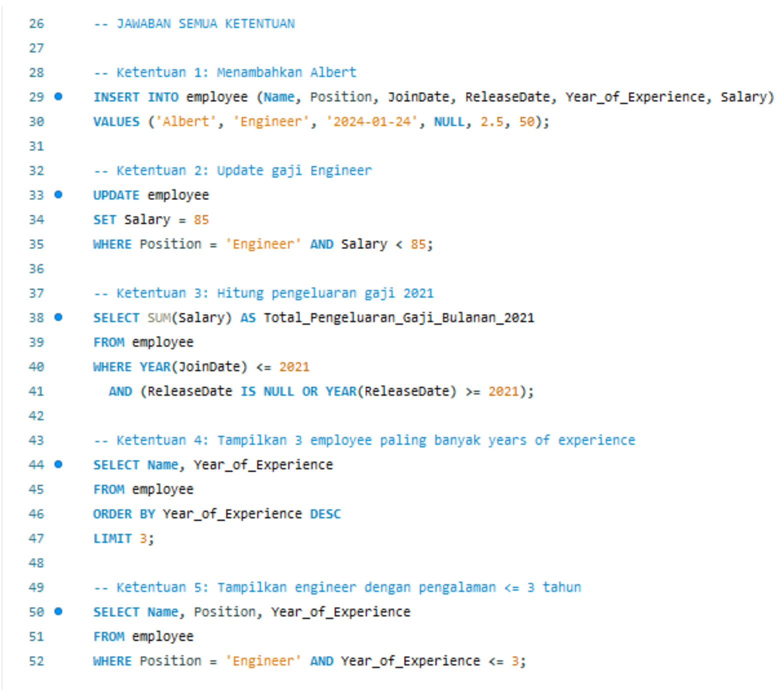
How to Run

1. Open your SQL Client: Launch MySQL Workbench or preferred SQL Client.
2. Open the Script File: Open the SQL_Script.sql file in your client's query editor.
3. Execute the Entire Script

CODE



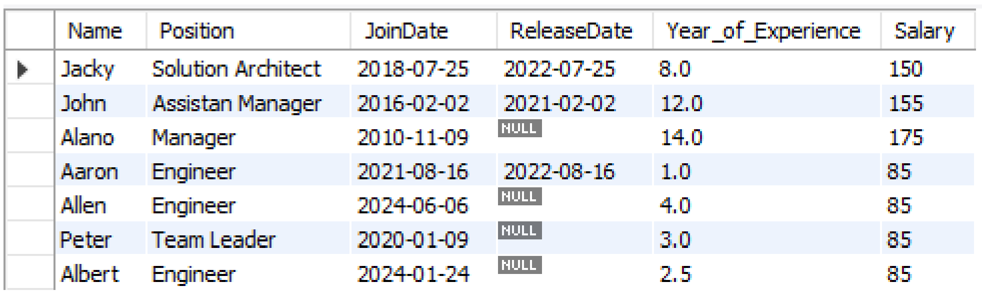
```sql
-- Buat database
CREATE DATABASE perusahaan_db;

-- Pilih database yang akan digunakan
USE perusahaan_db;

-- Membuat Tabel
CREATE TABLE employee (
    Name VARCHAR(50),
    Position VARCHAR(50),
    JoinDate DATE,
    ReleaseDate DATE,
    Year_of_Experience DECIMAL(4,1),
    Salary INT
);

-- Memasukkan data awal
INSERT INTO employee (Name, Position, JoinDate, ReleaseDate, Year_of_Experience, Salary) VALUES
('Jacky', 'Solution Architect', '2018-07-25', '2022-07-25', 8.0, 150),
('John', 'Assistan Manager', '2016-02-02', '2021-02-02', 12.0, 155),
('Alano', 'Manager', '2010-11-09', NULL, 14.0, 175),
('Aaron', 'Engineer', '2021-08-16', '2022-08-16', 1.0, 80),
('Allen', 'Engineer', '2024-06-06', NULL, 4.0, 75),
('Peter', 'Team Leader', '2020-01-09', NULL, 3.0, 85);
```



```sql
-- JAWABAN SEMUA KETENTUAN

-- Ketentuan 1: Menambahkan Albert
INSERT INTO employee (Name, Position, JoinDate, ReleaseDate, Year_of_Experience, Salary)
VALUES ('Albert', 'Engineer', '2024-01-24', NULL, 2.5, 50);

-- Ketentuan 2: Update gaji Engineer
UPDATE employee
SET Salary = 85
WHERE Position = 'Engineer' AND Salary < 85;

-- Ketentuan 3: Hitung pengeluaran gaji 2021
SELECT SUM(Salary) AS Total_Pengeluaran_Gaji_Bulanan_2021
FROM employee
WHERE YEAR(JoinDate) <= 2021
  AND (ReleaseDate IS NULL OR YEAR(ReleaseDate) >= 2021);

-- Ketentuan 4: Tampilkan 3 employee paling banyak years of experience
SELECT Name, Year_of_Experience
FROM employee
ORDER BY Year_of_Experience DESC
LIMIT 3;

-- Ketentuan 5: Tampilkan engineer dengan pengalaman <= 3 tahun
SELECT Name, Position, Year_of_Experience
FROM employee
WHERE Position = 'Engineer' AND Year_of_Experience <= 3;
```

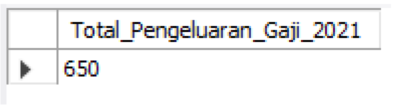
OUTPUT

- 1 and 2



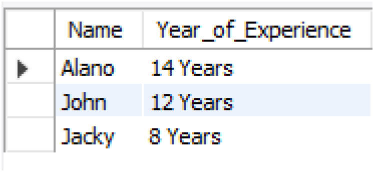
Name	Position	JoinDate	ReleaseDate	Year_of_Experience	Salary
Jacky	Solution Architect	2018-07-25	2022-07-25	8.0	150
John	Assistan Manager	2016-02-02	2021-02-02	12.0	155
Alano	Manager	2010-11-09	NULL	14.0	175
Aaron	Engineer	2021-08-16	2022-08-16	1.0	85
Allen	Engineer	2024-06-06	NULL	4.0	85
Peter	Team Leader	2020-01-09	NULL	3.0	85
Albert	Engineer	2024-01-24	NULL	2.5	85

- Task 3



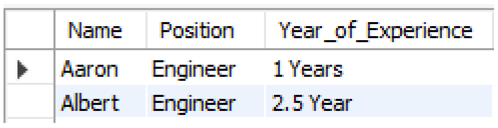
Total_Pengeluaran_Gaji_2021
650

- Task 4



Name	Year_of_Experience
Alano	14 Years
John	12 Years
Jacky	8 Years

- Task 5



Name	Position	Year_of_Experience
Aaron	Engineer	1 Years
Albert	Engineer	2.5 Year

Attachments

- Server.js : It is the main script that Node.js executes to create, configure, and run your entire web server.
- Collect_data.sh & cleanup_data.sh : The script for automation testing
- Cron_yyyy-mm-dd_hhmm.csv : automation testing output file
- SQL_Scripts.sql : The script for Data Processing