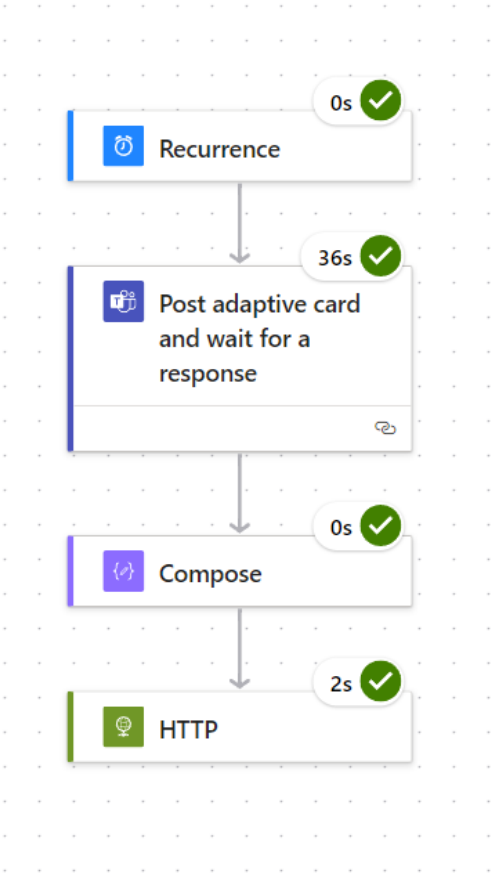
Teams Bot

Making the bot in Power Automate

I created a bot that is triggered once a day and sends a multiple-choice questionnaire to a Teams chat.

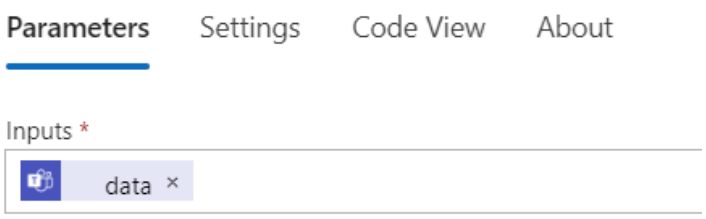


The post adaptive card action displays the questions to the user in Teams. The Compose action gets the answer data from the user. The HTTP action sends the answer data to the backend.

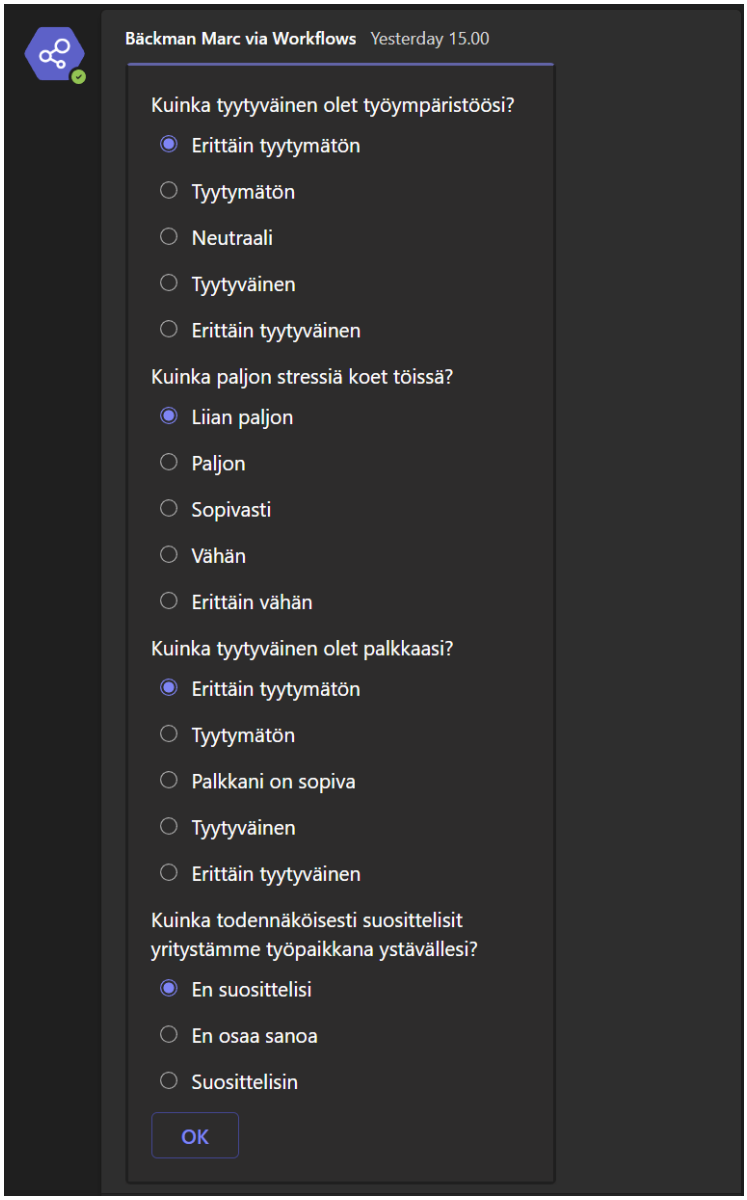
The data inside the adaptive card is saved like in this form:

```
"data": {
  "workPlace": "",
  "stress": "",
  "salary": "",
  "recommendation": ""
}
```

and is then collected by the Compose action.

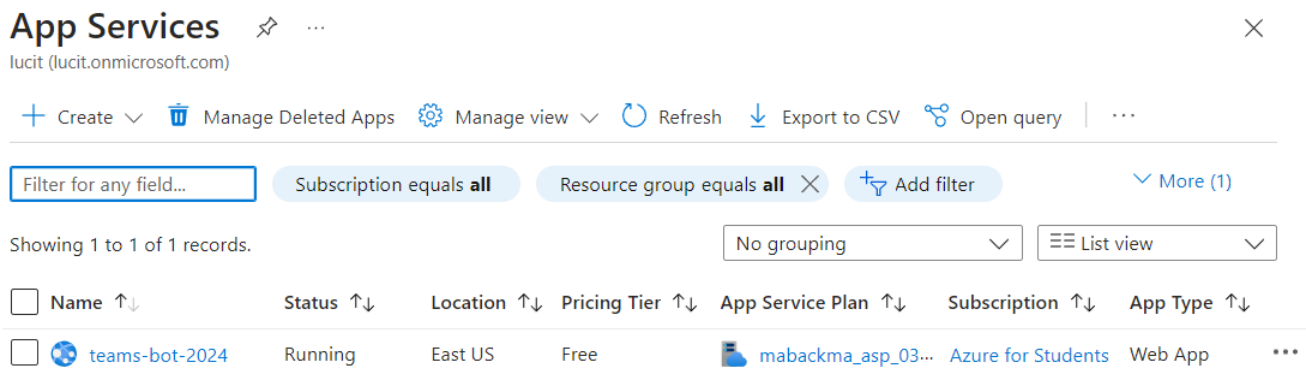


The questionnaire in Teams:



The backend for collecting the data

I made the backend with Python Flask. It collects the data that is sent from the HTTP action in Power Automate and will then send the data to a database. The backend is deployed at Microsoft Azure.



Backend repository: https://github.com/mabackma/TeamsBot

The pipeline

I made the pipeline directly from Azure Command-Line Interface (CLI), where I initially cloned the repository. I used these commands to make the pipeline with my repository:

 # Navigate to your App Service resource

az webapp show --name <app-name> --resource-group <resource-group-name>

 # Set up continuous deployment from GitHub

az webapp deployment source config --name <app-name> --resource-group <resource-group-name> --repo-url <github-repo-url> --branch <branch-name> --git-token <github-personal-access-token>

```
mabackma [ ~ ]$ az webapp deployment source config --name teams-bot-2024 --resource-group mabackma_rg_3183     --repo-url https://github.com/mabackma/Tea
msBot.git --branch master --git-token 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
location is not a known attribute of class <class 'azure.mgmt.web.v2023_01_01.models._models_py3.SourceControl'> and will be ignored
source_control_name is not a known attribute of class <class 'azure.mgmt.web.v2023_01_01.models._models_py3.SourceControl'> and will be ignored
location is not a known attribute of class <class 'azure.mgmt.web.v2023_01_01.models._models_py3.SiteSourceControl'> and will be ignored
{
  "branch": "master",
  "deploymentRollbackEnabled": false,
  "gitHubActionConfiguration": null,
  "id": "/subscriptions/cf6cb00a-2818-4a71-b399-04f516d38869/resourceGroups/mabackma_rg_3183/providers/Microsoft.Web/sites/teams-bot-2024/sourcecontrols/
web",
  "isGitHubAction": false,
  "isManualIntegration": false,
  "isMercurial": false,
  "kind": null,
  "location": "East US",
  "name": "teams-bot-2024",
  "repoUrl": "https://github.com/mabackma/TeamsBot",
  "resourceGroup": "mabackma_rg_3183",
  "type": "Microsoft.Web/sites/sourcecontrols"
}
```

The command explained:

"az webapp deployment source config": We are configuring the deployment source for the web app

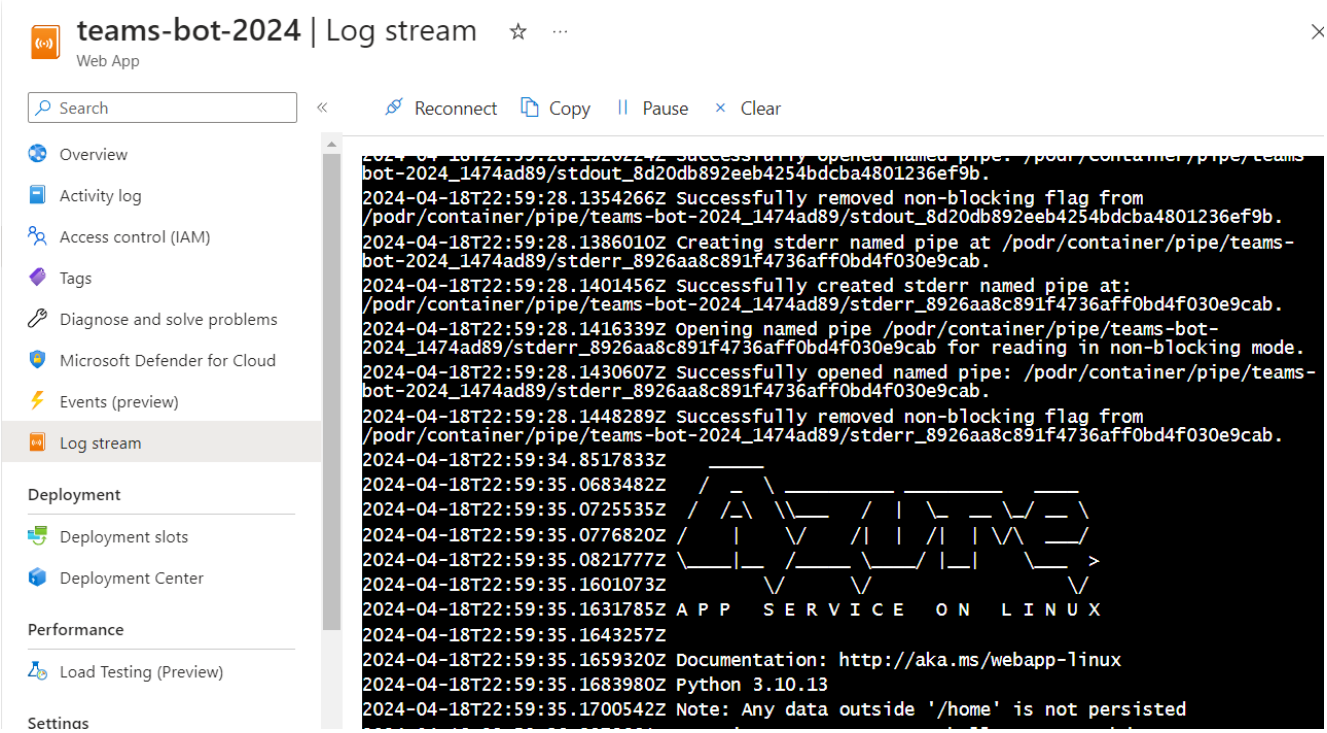
"—name teams-bot-2024": the name of the Azure Web App

"—resource-group mabackma_rg_3183": the name of the resource group (container that holds resources for the application).

"—repo-url https://github.com/mabackma/TeamsBot.git": the GitHub address of the project.

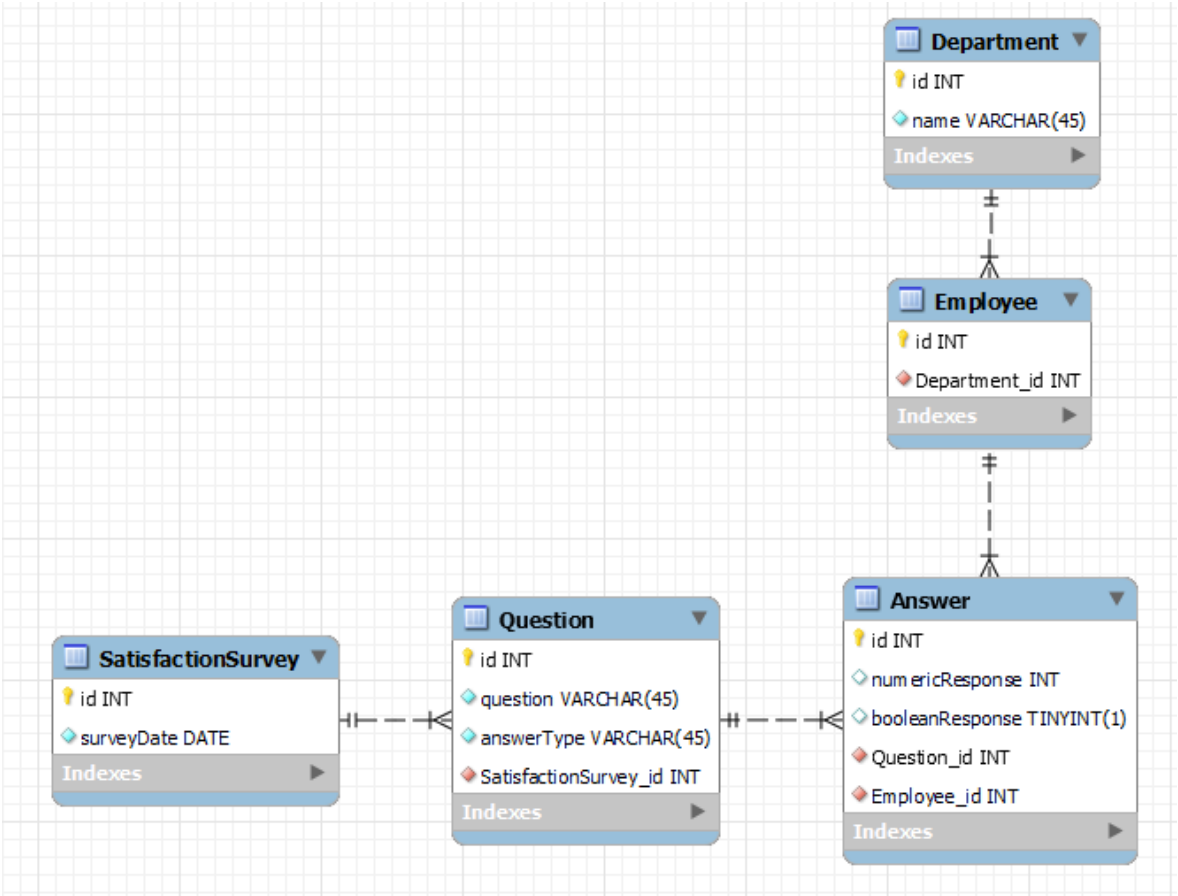
"—branch master": the branch used for deployment

I made a couple of changes to my repository and saw that the webapp in Azure rebooted after I pushed my changes to GitHub. The back end can be watched from looking at the log stream:



The database

I created the database schema using mySQL Workbench and generated a mySQL query for creating the schema.

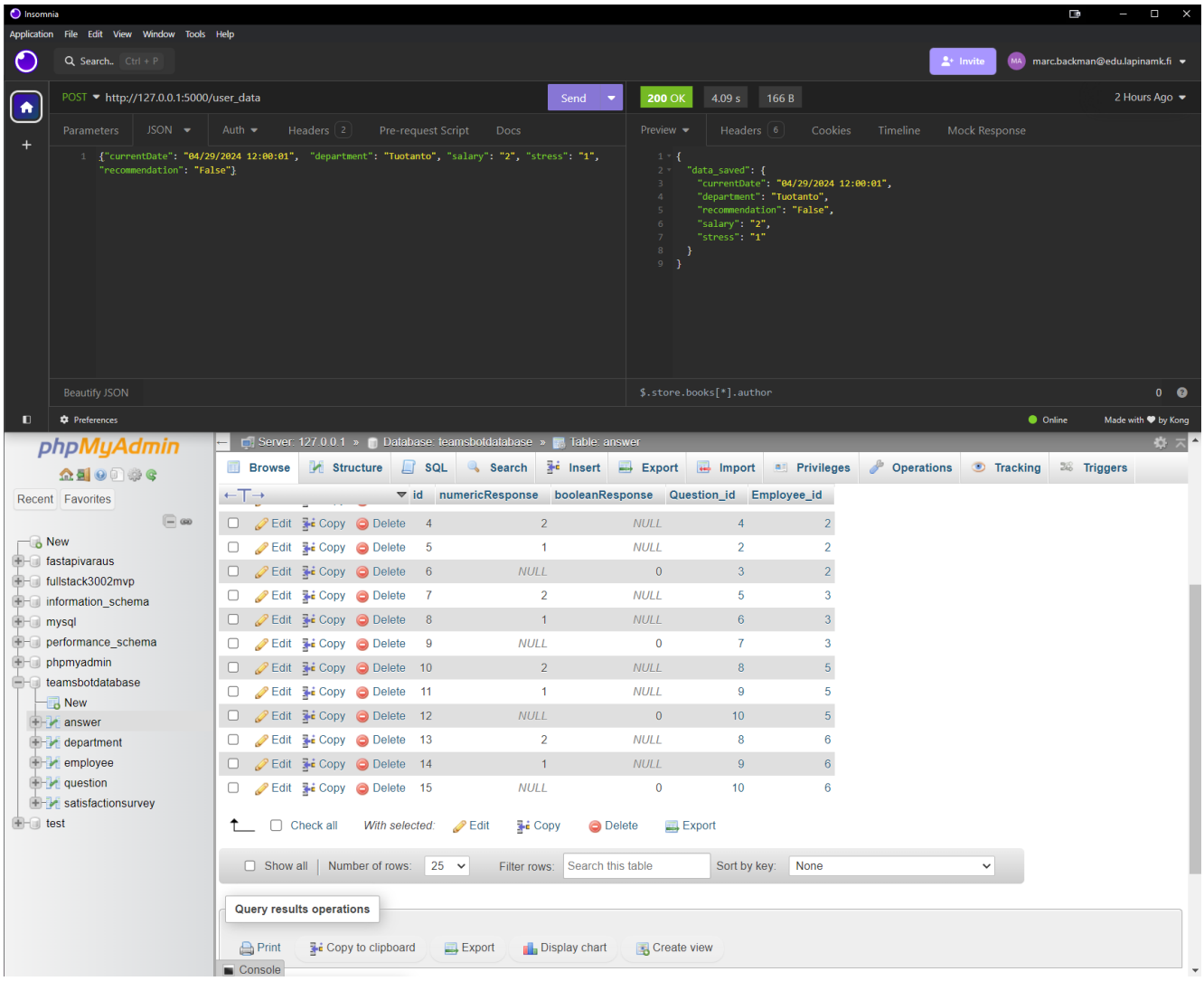


```sql
-- -----------------------------------------------------
-- Schema teamsBotDatabase
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `teamsBotDatabase` DEFAULT CHARACTER SET utf8 ;
USE `teamsBotDatabase` ;

-- -----------------------------------------------------
-- Table `teamsBotDatabase`.`Department`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teamsBotDatabase`.`Department` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  UNIQUE INDEX `name_UNIQUE` (`name` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `teamsBotDatabase`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teamsBotDatabase`.`Employee` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Department_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_Employee_Department1_idx` (`Department_id` ASC),
  CONSTRAINT `fk_Employee_Department1`
    FOREIGN KEY (`Department_id`)
    REFERENCES `teamsBotDatabase`.`Department` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `teamsBotDatabase`.`SatisfactionSurvey`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teamsBotDatabase`.`SatisfactionSurvey` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `surveyDate` DATE NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  UNIQUE INDEX `surveyDate_UNIQUE` (`surveyDate` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `teamsBotDatabase`.`Question`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teamsBotDatabase`.`Question` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `question` VARCHAR(45) NOT NULL,
  `answerType` VARCHAR(45) NOT NULL,
  `SatisfactionSurvey_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_Question_SatisfactionSurvey_idx` (`SatisfactionSurvey_id` ASC),
  CONSTRAINT `fk_Question_SatisfactionSurvey`
    FOREIGN KEY (`SatisfactionSurvey_id`)
    REFERENCES `teamsBotDatabase`.`SatisfactionSurvey` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `teamsBotDatabase`.`Answer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `teamsBotDatabase`.`Answer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `numericResponse` INT NULL,
  `booleanResponse` TINYINT(1) NULL,
  `Question_id` INT NOT NULL,
  `Employee_id` INT NOT NULL,
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  PRIMARY KEY (`id`),
  INDEX `fk_Answer_Question1_idx` (`Question_id` ASC),
  INDEX `fk_Answer_Employee1_idx` (`Employee_id` ASC),
  CONSTRAINT `fk_Answer_Question1`
    FOREIGN KEY (`Question_id`)
    REFERENCES `teamsBotDatabase`.`Question` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Answer_Employee1`
    FOREIGN KEY (`Employee_id`)
    REFERENCES `teamsBotDatabase`.`Employee` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Testing the database

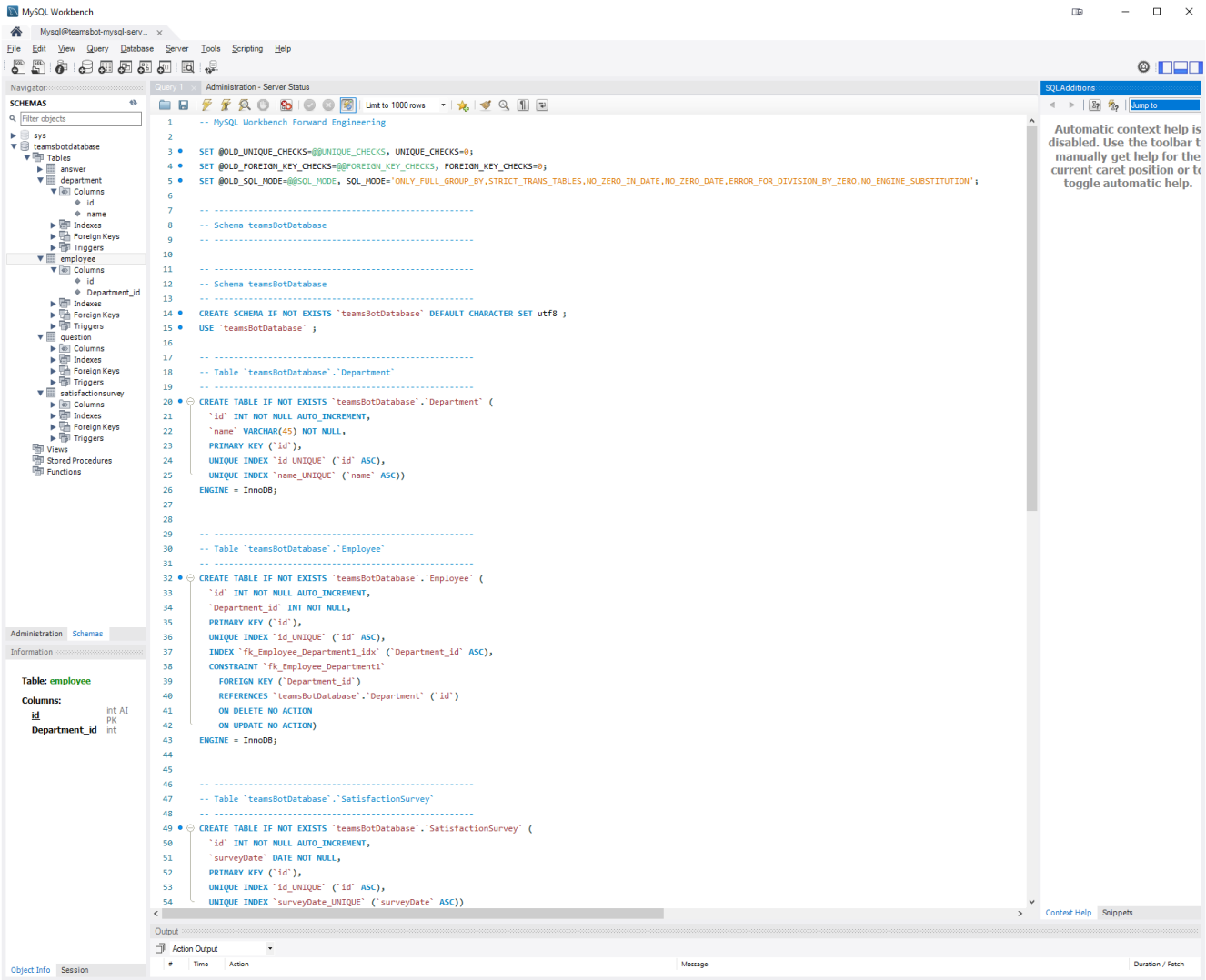
I tested the database with XAMPP and phpMyAdmin. This graphical user interface was useful to create the tables and see if my code would save data in the tables.

I made models for the tables in my mySQL schema to the Pyhon Flask backend using SQLAlchemy. To test that my models were working correctly, I used Insomnia to send data to the backend and phpMyAdmin to see if it was saved in the database.



Creating the database in Azure

After verifying that everything between the backend and the mySQL database is working correctly, I created the database in a mySQL server in Azure through using mySQL Workbench.



I also added a connection string into my Python code for connecting to the mySQL database at Azure instead of the mySQL database at phpMyAdmin.

The connection string looks like this:

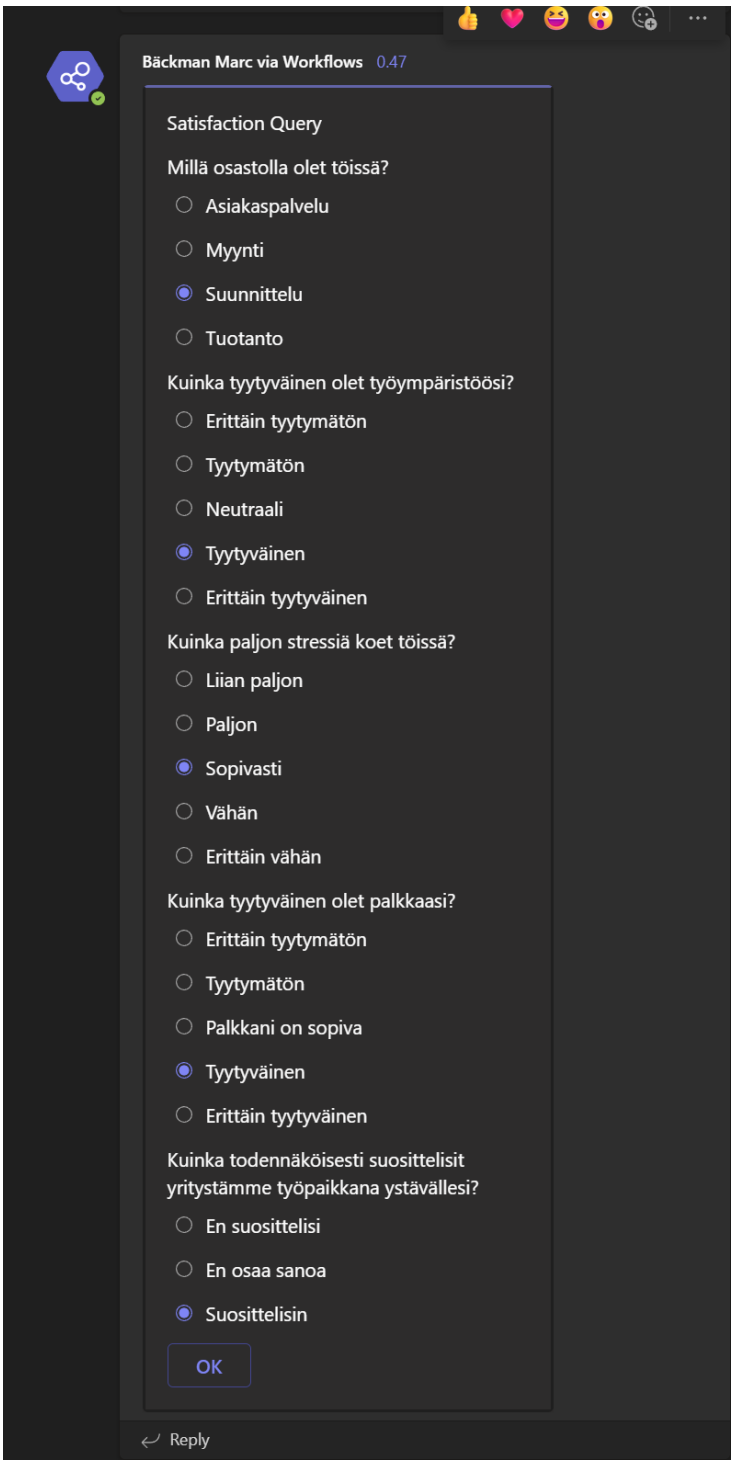
mysql+mysqlconnector://{azure_mysql_username}:{azure_mysql_password}@{azure_mysql_host}:{azure_mysql_port}/{azure_mysql_database}

I added environment variables for the connection string to the App service teams-bot-2024 in Azure.

Testing the workflow

I then tested the whole workflow by sending a survey from Power Automate to teams, answering the survey in teams, watching the log stream of the app service, and verifying the data saved in the mySQL database at Azure. Everything was working correctly.

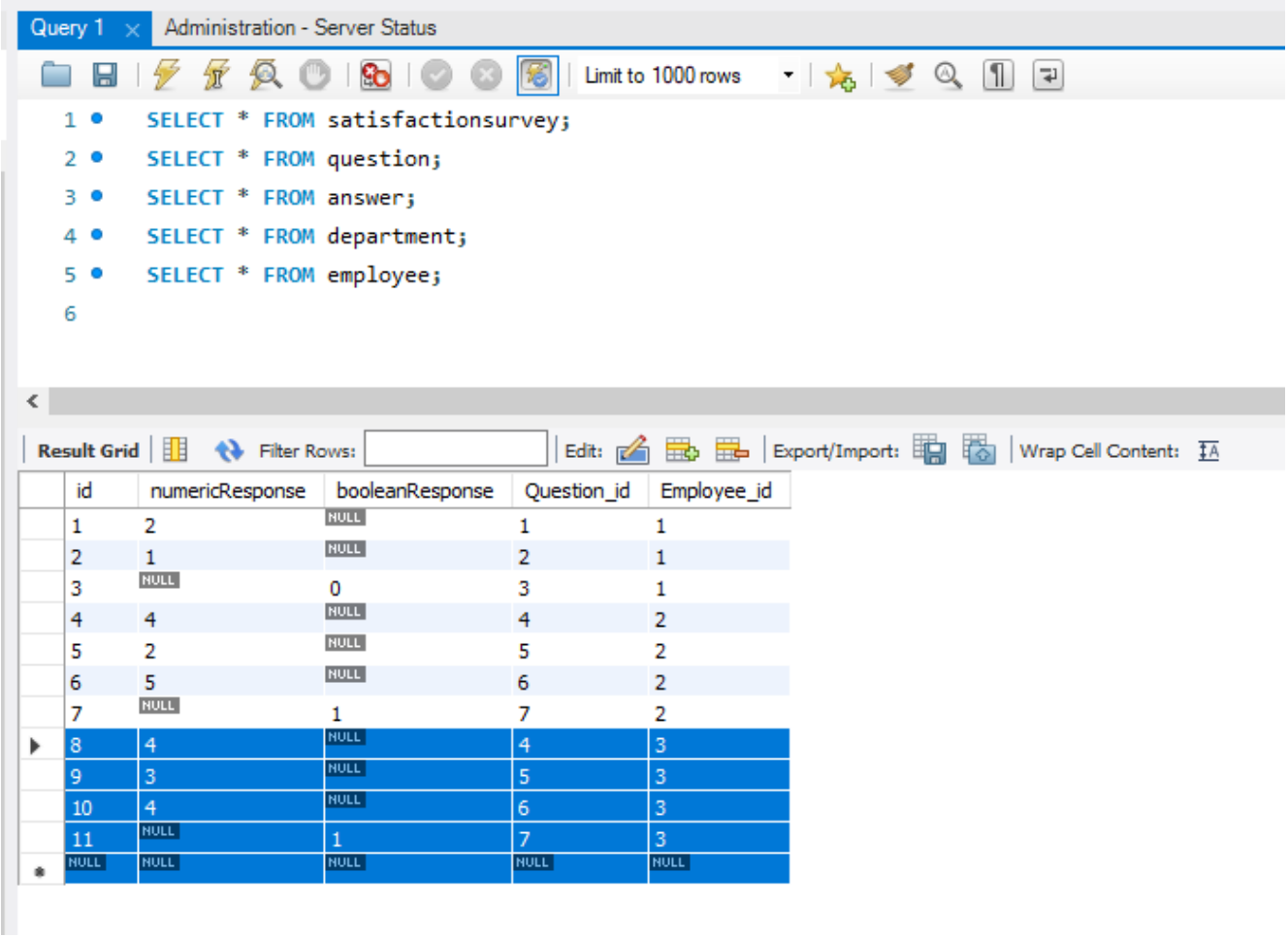
In teams:



After pressing OK button, in teams-bot-2024 log stream:

```
2024-04-28T21:46:41  No new trace in the past 16 min(s).
2024-04-28T21:47:41  No new trace in the past 17 min(s).
2024-04-28T21:48:41  No new trace in the past 18 min(s).
2024-04-28T21:49:29.3476579Z data received: {'currentDate': '04/28/2024 21:47:23',
'department': 'Suunnittelu', 'workPlace': '4', 'stress': '3', 'salary': '4',
'recommendation': 'True'}
2024-04-28T21:49:29.4027641Z 169.254.129.1 - - [28/Apr/2024:21:49:29 +0000] "POST
/user_data HTTP/1.1" 200 146 "-" "azure-logic-apps/1.0 (workflow
f191a87be1504345b7561f402a1b4015; version 08584877830800881858) microsoft-flow/1.0"
```

At mySQL datase from making query at mySQL Workbench:



id	numericResponse	booleanResponse	Question_id	Employee_id
1	2	NULL	1	1
2	1	NULL	2	1
3	NULL	0	3	1
4	4	NULL	4	2
5	2	NULL	5	2
6	5	NULL	6	2
7	NULL	1	7	2
8	4	NULL	4	3
9	3	NULL	5	3
10	4	NULL	6	3
11	NULL	1	7	3
NULL	NULL	NULL	NULL	NULL

The workflow was tested a few times and then I concluded that it was working properly. I then started making a Streamlit application for displaying the data that was in the database. I should mention that I had to set the public ip address of my computer to the database in order to connect to it from mySQL Workbench.

Streamlit

I made a Streamlit application to show the results in different charts. Getting the results from the database had to be done through the backend service application that was deployed in Azure, because the mySQL database requires the public ip address for connecting to the database.

I wrote queries for getting data from the database. The queries were sent to the app service in Azure and then I made different Streamlit charts to display the data. I wanted to display data so that you could compare the answers from different departments.The nice thing about Streamlit charts is that they are interactive. For example, you can press on a department and remove it from the chart. Hovering your mouse over a department on a chart will also give you more information on the department.

The queries

Query all answers and deparment:

```
SELECT answer.id, department.name AS department_name
FROM answer
JOIN employee ON answer.employee_id = employee.id
JOIN department ON employee.department_id = department.id
```

Query answers over time with department and satisfactionsurvey:

```
SELECT satisfactionsurvey.surveyDate, department.name AS department_name
FROM answer
JOIN question ON answer.question_id = question.id
JOIN satisfactionsurvey ON question.satisfactionsurvey_id = satisfactionsurvey.id
JOIN employee ON answer.employee_id = employee.id
JOIN department ON employee.department_id = department.id
```

Query recommendation from everyone:

```
SELECT * FROM answer WHERE Question_id IN (SELECT id FROM question WHERE question = 'recommendation' GROUP BY id)
```

Query average satisfaction with salary and department:

```
SELECT AVG(answer.numericResponse) AS avg_rating_salary, department.name AS department_name
FROM answer
JOIN employee ON answer.employee_id = employee.id
JOIN department ON employee.department_id = department.id
WHERE answer.Question_id IN (SELECT id FROM question WHERE question = 'salary')
GROUP BY department_name
```

Query average satisfaction with work place and department:

```
SELECT AVG(answer.numericResponse) AS avg_rating_workplace, department.name AS department_name
FROM answer
JOIN employee ON answer.employee_id = employee.id
JOIN department ON employee.department_id = department.id
WHERE answer.Question_id IN (SELECT id FROM question WHERE question = 'workPlace')
GROUP BY department_name
```

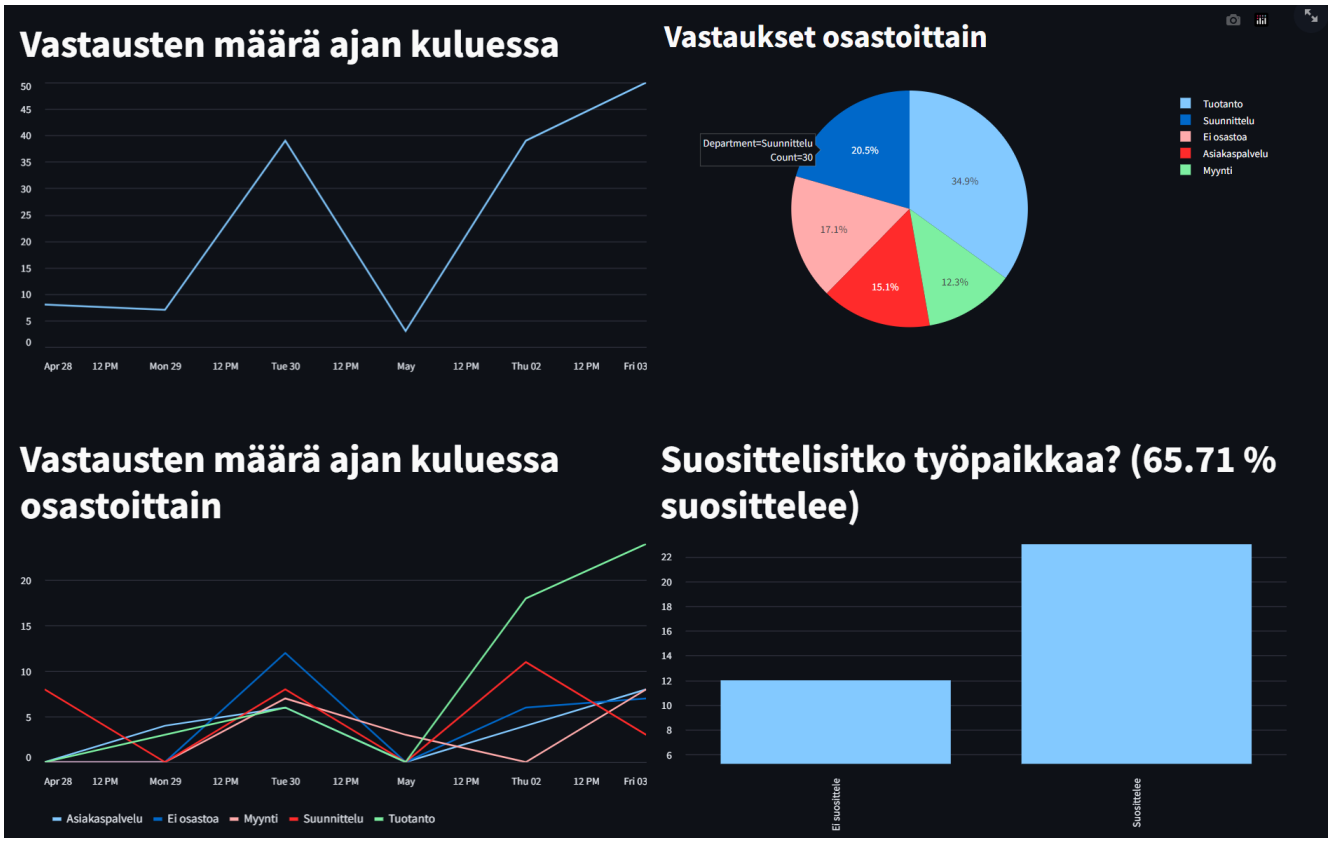
Query average satisfaction with stress and department:

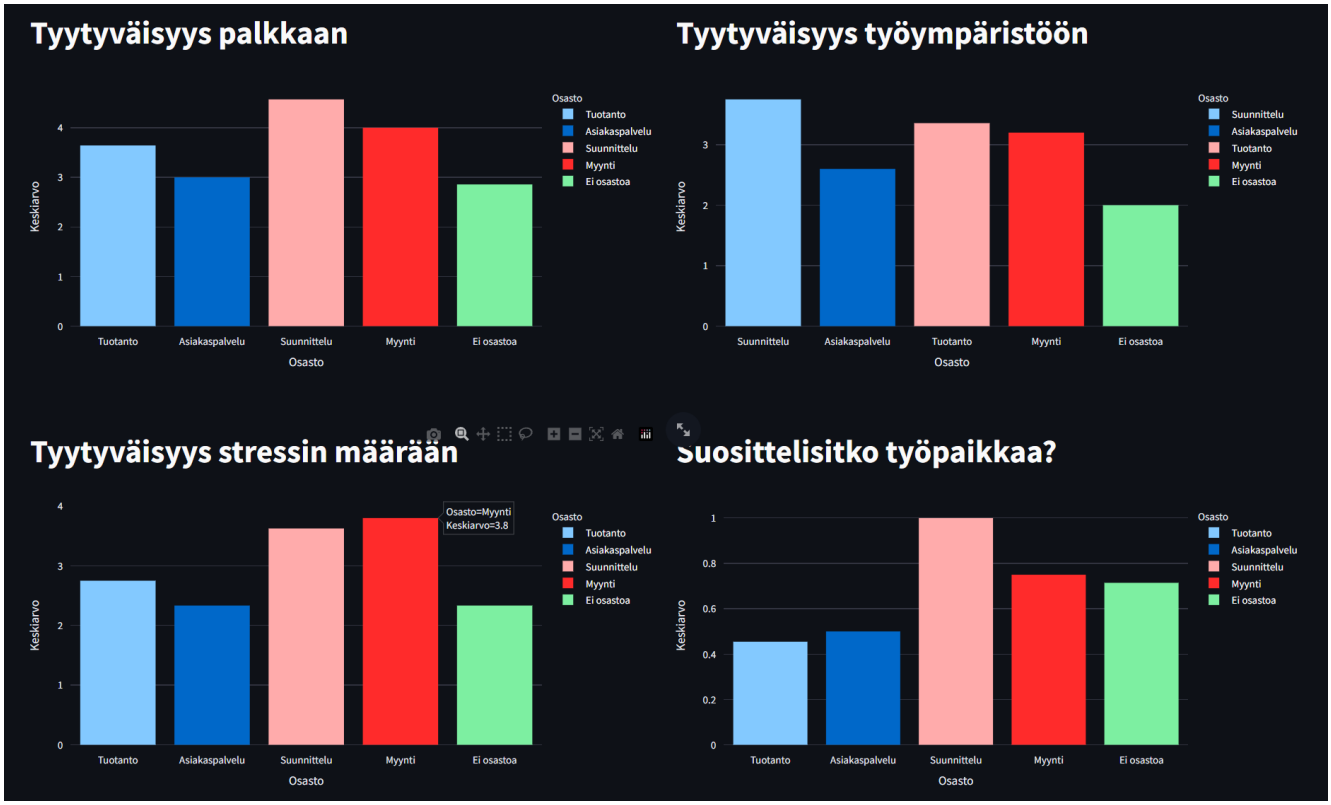
```
SELECT AVG(answer.numericResponse) AS avg_rating_stress, department.name AS department_name
FROM answer
JOIN employee ON answer.employee_id = employee.id
JOIN department ON employee.department_id = department.id
WHERE answer.Question_id IN (SELECT id FROM question WHERE question = 'stress')
GROUP BY department_name
```

Query average recommendation and department:

```
SELECT AVG(answer.booleanResponse)
AS avg_rating_recommendation, department.name
AS department_name
FROM answer
JOIN employee ON answer.employee_id = employee.id
JOIN department ON employee.department_id = department.id
WHERE answer.Question_id IN (SELECT id FROM question WHERE question = 'recommendation')
GROUP BY department_name
```

The charts





GitHub repositories:

Teams Bot app service: https://github.com/mabackma/TeamsBot

Streamlit application deployed at: https://teams-bot.streamlit.app/

Teams Bot Demo: https://www.youtube.com/watch?v=TZwx1xhPjZI

Conclusion

The workflow for the bot works as expected and the results are updated in a public Streamlit application. The Streamlit application is an effective way to monitor changes in the results in real time. However, the adaptive card for the survey should be sent to individual people in Teams instead of a channel. This is an easy fix that can be done in the Power Automate Flow.

Another improvement that could be done is using the NPS (Net Promoter Score) standard for a better calculation of the employees' satisfaction.

Marc Bäckman

REFERENCES:

Azure SQL Database Tutorial | Relational databases in Azure,
https://www.youtube.com/watch?v=BgvEOkcR0Wk

Develop applications with Azure Database for MySQL - Flexible Server,
https://learn.microsoft.com/en-us/training/modules/develop-apps-with-azure-database-mysql/