**Pollinate – Service Management Tech Assignment** **- DevOps**

**Goal**

1. Verify the candidate's ability to take mid-level requirements and deliver a working solution.
2. Verify the candidate's ability to create a maintainable environment with different services.
3. Verify the candidate's ability to automate a real-world use case.
4. Verify the candidate's ability to communicate and present technical solutions.

**Deliverables**

1. Candidates needs to deliver the solution to the assignment.
   1. High level design document (pdf)
   2. Proof of concept (code)
2. Public GitHub repository with all associated code, comments, documentation, and diagrams related to the assignment for all parts of the assignment attempted.

Note: Arrange the GitHub repository by sections in the question paper.

**Notes**

1. Provide as much detail as you feel necessary, including documentation and diagrams as you will be asked to present this back to us as part of the final stage of the process.
   1. Test is open book.
2. Please complete the assignment within 72 hours of receipt to the best of your abilities with the time constraints.

Assignment

**Objective**

The objective of this exercise is to determine the applicant’s grasp of certain technologies.

**Assignment**

**Section 1:**

1. Create a proof of concept of the following:
   * Create an application with simple API that will be used to submit a curl command (curl -X POST http:// /app) that will insert the date-time stamp into your database.
2. The proof of concept should consist of a single web-application and a single database.
3. Create a GitHub project for this assignment.
4. Automation the provisioning your database node and web node with the API you created.
5. A readme file must be populated with your git project detailing your solution and how to run it.
6. Provide details of your high-level design that would meet the requirements outlined above.
   * This design must be submitted 24 hours prior to the interview via pdf and must show a logical design.

**Additional Requirements:**

1. The service will accept a POST request that will persist the timestamp of that event into a database. For all aspects of your solution, you should consider multi-DC, zones & redundancy of components.
2. Please note that the design must be elastic based on demand.
3. For any areas, which you have not considered to automate, please state these clearly and if possible, provide details of the solution you would have implemented including your experience/depth of knowledge of that area.
4. Be ready to explain your reasons regarding your choice of solution.
5. The choice of technology is up to you. For example, if you choose to use a NoSQL database or a Relational database is your choice.
6. The provisioning tool suggested below is Ansible, however, you may use any tool of your choice for your automation.

**Questions and Tasks:**

1. Explain with as much detail as you can of your high-level design and explain why that would meet the requirements outlined above.
2. Provide details of your web API that will be used to submit a curl command (curl -X POST http:// /app) that will insert the Date/Time stamp into your database. The API code should be well documented.
3. In addition, provide any automation for your backup/restore process you would implement.
4. Provide details of your Persistency Layer, with details of your cluster setup and configuration.
5. If you consider any Load Balancer or queuing service to be used, please be ready to explain the reasons and your suggested configurations for each.
6. Please consider monitoring and maintainability.
7. Please explain how elastic your service is, what would be the trigger points and how the scale up or down would work.

**Section 2:**

**Questions related to Database programming**. (Prefer to attempt in MSSQL, include the DDL, DMLs as .sql files.). Free to prepare any documentation if necessary.

1. Query all columns for a city in **CITY** with the *ID* 1661.

The **CITY** table is described as follows:

Table

Description automatically generated

2. Write a query that prints a list of employee names (i.e.: the *name* attribute) for employees in **Employee** having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending *employee\_id*.

**Input Format**

The **Employee** table containing employee data for a company is described as follows:

Table

Description automatically generated

where *employee\_id* is an employee's ID number, *name* is their name, *months* is the total number of months they've been working for the company, and *salary* is their monthly salary.

**Sample Input**

Table

Description automatically generated

**Sample Output**

Graphical user interface, application

Description automatically generated

**Explanation**

Graphical user interface, text, application

Description automatically generated

3.Generate the following two result sets:

Query an *alphabetically ordered* list of all names in **OCCUPATIONS**, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).

Query the number of ocurrences of each occupation in **OCCUPATIONS**. Sort the occurrences in *ascending order*, and output them in the following format:



where [occupation\_count] is the number of occurrences of an occupation in **OCCUPATIONS** and [occupation] is the *lowercase* occupation name. If more than one *Occupation* has the same [occupation\_count], they should be ordered alphabetically.

**Note:** There will be at least two entries in the table for each type of occupation.

The **OCCUPATIONS** table is described as follows:

Table

Description automatically generated

*Occupation* will only contain one of the following values: **Doctor**, **Professor**, **Singer** or **Actor**.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| An **OCCUPATIONS** table that contains the following records: |  |
| Table  Description automatically generated | Ashely(P)  Christeen(P)  Jane(A)  Jenny(D)  Julia(A)  Ketty(P)  Maria(A)  Meera(S)  Priya(S)  Samantha(D)  There are a total of 2 doctors.  There are a total of 2 singers.  There are a total of 3 actors.  There are a total of 3 professors. |

**Explanation**

The results of the first query are formatted to the problem description's specifications.  
The results of the second query are ascendingly ordered first by number of names corresponding to each profession (2<= 2<= 3<=3), and then alphabetically by profession doctor<= singer and actor <= professor)

4.  *P(R)* represents a pattern drawn by Julia in *R* rows. The following pattern represents *P(5)*:

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

Write a query to print the pattern *P(20)*.

*5. P(R)* represents a pattern drawn by Julia in *R* rows. The following pattern represents *P(5)*:

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

Write a query to print the pattern *P(20)*.

6. Write a (dynamic) PIVOT to obtain the below result:

Use the DML and DDL statements below.

Create table Projects

(

ProjectID int

,[ProjectedMonth] Date

,Labor Int

,Equipment int

)

Insert into Projects values (1,'2021-09-01', 500, 0)

, (1,'2021-10-01', 250,250)

, (1,'2021-11-01', 100,50)

, (2,'2021-11-01', 100,50)

, (2,'2021-12-01', 100,50)

**Expected output:**

Table

Description automatically generated

**Section 3: Applies to Data platform Candidates**

Create a Data factory instance to ingest data to Azure SQL database Temporary Table in your SQL Azure SQL database.

You are free to use Azure Free account for this.

Step 1: Data Ingestion.

Source link: [Hollywood Theatrical Market Synopsis 1995 to 2021 | Kaggle](https://www.kaggle.com/johnharshith/hollywood-theatrical-market-synopsis-1995-to-2021?select=HighestGrossers.csv)

Dataset: HighestGrossers.csv

Hint: You can use HTTP linked service for this but not limited.

Prefer to use Azure Key vault for storing and accessing sensitive data such as Database credentials.

Database Name: DataMart

Schema: temp\_stage

Table: landing\_MovieDetails

Step 2: Modelling section. Free to decide the columns for the tables mentioned below.

Once data is ingested to the Temp Table above, model the data in to the below mentioned fact and dimension table. Use a simple modelling and insert the data into the below tables.

Note: Free to use any functionality in Azure Data factory, Expectation is a simple ingestion of CSV into temporary table in Azure SQL database.

Schema: dw

Table: Create two tables,

Fact: Fact\_moviestatistics

Dimension: DI\_movie

Step 3: Reporting Section

1. Create a stored procedure to get the Total tickets sold (Highest – lowest) by Genre

2. Create a stored procedure to generate a report to get the movie name, Genre, Year, Rating, Total for Year.

Result of the stored procedure should be in descending order of the column [Total\_for\_year]

Note: report is not required. Result of the stored procedure is what the expectation.

Provide a readme file for the solution with relevant details and approaches.