SJSU SAN JOSÉ STATE UNIVERSITY

Department of Applied Data Science

DATA 225

# Database Systems for Analytics

Instructor: Simon Shim

LAB GROUP PROJECT REPORT 1

LinkedIn Job Postings

Group 4

Group Members

Anshul Yadav

Sakshi Jain

Shreya Sree Matta

Veena Ramesh Beknal

## INDEX

**PROBLEM STATEMENT:** 3

**SOLUTION REQUIREMENTS:** 3

**LIMITATIONS:** 3

**CONCEPTUAL DATABASE DESIGN:** 4

**ENTITY RELATIONSHIP DIAGRAM:** 5

**FUNCTIONAL ANALYSIS:** 5

- **SQL CODE SNIPPETS/QUERIES:** 6
- **STORED PROCEDURE** 9
- **TRIGGERS** 10

**ACCESS PRIVILEGES** 11

**LOGGING OF DATABASE:** 11

- **AWS CONNECTIVITY WITH PYTHON:** 11

**PERFORMANCE MEASUREMENT:** 13

- ***AWS*** 13
- ***JMeter*** **Error! Bookmark not defined.**

**GITHUB REPO** **Error! Bookmark not defined.**

## PROBLEM STATEMENT:

In the post-pandemic job market, both job seekers and recruiters must navigate multiple obstacles. There is a critical need for candidates to understand job trends, skills, and compensation trends to make informed discussions while recruiters need to attract the right talent that fits the company culture with the right skills and right compensation. Various platforms are available for job listings, but LinkedIn is the primary platform for networking and job postings since it has close to 900 million members on the platform and offers a telescopic view of job-related information along with company and individual updates. Using the dataset "LinkedIn Job Posting Analysis" available on Kaggle provides an extensive collection of job listings from the platform for 2023. This dataset contains a nearly comprehensive record of 15,000+ job postings listed over the course of 2 days. Each individual posting contains 27 valuable attributes, including the title, job description, salary, location, application URL, and work types (remote, contract, etc.). In addition to separate files containing the benefits, skills, and industries associated with each posting. The majority of jobs are also linked to a company, which are all listed in another CSV file containing attributes such as the company description, headquarters location, number of employees, and follower count.

Data from LinkedIn is a rich source for job seekers to identify industry hiring trends and recruiters to find the ideal candidates. Storing all these data points in a structured manner like a database with real-time updates can be invaluable and the potential for exploration of this dataset is vast. This database can be used to identify current open job positions, in-demand skills, compensation trends, companies/industries primed for growth, etc.

Analysis of data from this database combined with other datasets can also offer macro and micro trends of the job market scenario and aid job seekers in improving their respective careers. Connecting the database to a cloud platform, Amazon AWS in this case, ensures reliability, scalability, and consistency, and ensures that the access is location-agnostic.

## SOLUTION REQUIREMENTS:

We propose a robust Relational Database Management System (RDBMS) architecture for storing the multiple datasets available from LinkedIn. RDBMS is a structured way to manage, store, and retrieve large volumes of data. The database will be created in MySQL and Amazon RDS to facilitate querying and analysis. This database facilitates in-depth analysis, ensures data integrity, and empowers job seekers and companies to make better-informed decisions in this competitive job market. We have also illustrated some key use case queries to show how to navigate the data and answer questions that would typically exist in the mind of an individual entering the job market or someone looking to make a switch from an existing job.

## LIMITATIONS:

While this database is robust and can deliver as intended, some limitations must be addressed:

- Data integrity: The database is as good as the data it contains and since there is no mandate to update all the data fields, completeness of the data is not in our control. In several cases, the imputation of missing data would be incorrect since said imputation may not reflect the real world (like compensation)
- Normalization/standardization: Since job postings are posted by individuals

working for companies or hiring firms who post on behalf of the aforementioned firms, fields like country, city, state of the job may not be standardized (like NY, NYC, New York City for job location)

- Lack of historical data: Since the current data considered is a limited timeframe snapshot, historical trends, and comparisons aren't possible – this makes it challenging to understand if a current job market scenario is relatively normal or an outlier

## CONCEPTUAL DATABASE DESIGN:

To implement the LinkedIn Job Postings database, the database will require the below entities to store the data about the job postings and to retrieve job postings and company data.

1. Companies
2. Company Specialties
3. Companies Industries
4. Employee Counts
5. Job Postings
6. Benefits
7. Job Industries
8. Job skills

1. **Companies**: This entity stores the details about the companies that list jobs on LinkedIn and is unique at company_id level. This entity contains the following attributes:

   company_id, name, description, company_size, country_size, state, city, zip_code, address, url Primary Key: company_id

2. **Company Specialties:** This entity stores the details of specialties for companies. Since a single company can have multiple specialties, company_id -> specialty is a 1-to-many relationship.

   company_id, specialty
   Foreign Key: company_id

3. **Companies Industries:** This entity stores the details of industries for companies. Since a single company can be associated with multiple industries, company_id -> industry is a 1-to-many relationship.

   company_id, industry
   Foreign Key: company_id

4. **Employee Counts**: This entity stores the details of employee count and followers. Since this table contains a timestamp of when the headcount was recorded, a single company can have multiple employee_count and follower_count values.

   employee_count, company_id, follower_count, time_recorded
   Foreign Key: company_id

5. **Benefits:** This entity stores the details of benefits for jobs. Since a single job_id can have multiple benefits, job_id -> benefits is a 1-to-many relationship.

   job_id, inferred, type
   Foreign Key: job_id

6. **Job Industries:** This entity stores the industries for the jobs. Since a single job_id can be associated with multiple industry, job_id -> industry is a 1-to-many relationship.

   job_id, industry_id
   Foreign Key: job_id

7. **Job Skills:** This entity stores the skills required for the posted jobs. Since a single job_id can require multiple skills, job_id -> skill_abr is a 1-to-many relationship.

   job_id, skill_abr
   Foreign Key: job_id

8. **Job Postings:** This is the master dataset of job postings containing multiple columns pertaining to the posted job.
   Job_id, company_id, title, description, max_salary, med_salary, min_salary, pay_period, formatted_work_type, location, applies, original_listed_time, remote_allowed, views, job_posting_url, application_url, expiry, closed_time, formatted_experience_level, skills_desc, listed_time, posting_domain, sponsored, work_type, currency, compensation_type
   Primary Key: job_id

## ENTITY RELATIONSHIP DIAGRAM:

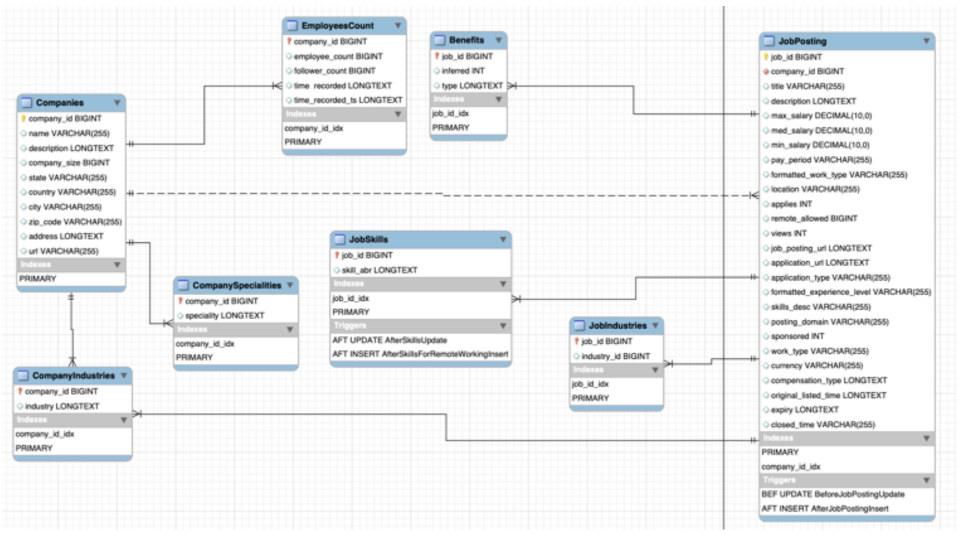


*Figure 1 ER Diagram*

## FUNCTIONAL ANALYSIS:

From the perspective of a job seeker, we can use the various entities detailed earlier, link them through queries and arrive at interesting data-driven insights to enhance one's professional career. For candidates, the key criteria while looking for a job are (in no particular order of priority) – company prospects, growth prospects, benefits, compensation, etc. We have listed multiple query snippets, stored procedures and triggers that can help a candidate evaluate the job market and take an informed decision. Below are some use cases for candidates to use this dataset and derive value:

- Average number of benefits per job and details of jobs offering higher than the average number of benefits – this can help in assessing which jobs are offering better than average benefits
- Find the average salary by each job industry; Rank to find the top 3 industries by avg salary – helps in identifying which industries are paying well and which aren't, ranking the top 3 gives the candidate a target of which are the top 3 industries to aim for, if compensation is a key factor
  - Top skills for

each job industry – this helps in narrowing down the top skills that would be required to get a job in those industries based
- Company employee headcount growth – this would help the candidate in identifying companies that have shown considerable growth over time indicating that the growth prospect is high
- Job openings by industry – this would help the candidate in picking industries where openings are higher since the likelihood of getting a job is higher, if the profile is a good match
- Which companies have the highest number of specialties by industry? – this helps in identifying top companies by specialty across each industry which helps the candidate to decide on which specialty is best suited for them
- What is the count of internships available with company name, size and job description? – this would help an undergrad or grad student in zeroing down on internship opportunities
- How many full-time jobs offer medical, dental, and 401k as benefits and have HQ in California? – this would help a candidate who's keen on looking for opportunities withing the state of California with specific benefits since the cost of living is high
- Average, maximum and minimum salaries for companies that are part of 'Nonprofit Organization Management' industry by each state – this would help candidates who are looking to choose from a range of opportunities in the non-profit industry across each state
- List of companies in a specific city and their average employee counts – this would help candidates get a quick snapshot of the companies and their respective headcount in a specific input city
- Finding companies with a specific specialty that also have job postings with a certain skill requirement – this would help candidates who are looking for niche opportunities i.e. very specific skillsets in specific companies/industries

## SQL CODE SNIPPETS/QUERIES:

[1] Average number of benefits per job and details of jobs offering higher than the average # of benefits

```
 ## Getting number of benefits in the first CTE
(Common Table Expression) ##

with num_benefits as (
          select job_id, count(distinct type) as
num_benefits
          from benefits
          group by 1
),
Getting avg number of benefits in the next CTE

avg_benefits as (
select avg(num_benefits) as avg_benefits_count
from num_benefits
)
select n.job_id, jp.company_id, c.name as
company_name, jp.title, num_benefits
from num_benefits n
inner join JobPosting jp
on n.job_id = jp.job_id
inner join Companies c
on jp.company_id = c.company_id
where num_benefits >= (select
avg_benefits_count from avg_benefits)
order by num_benefits desc;
```

[2] Find the average salary by each job industry. Rank to find the top 3 industries by avg salary (joins and window function)

```
## Getting avg of min_salary for all non-null
values by industry ##

with avg_salary_by_industry as (
select industry,  round(avg(min_salary),3) as
avg_salary
from JobPosting jp
inner join CompanyIndustries ci
on jp.company_id = ci.company_id
where industry is not null
and coalesce(min_salary,0) > 0
group by 1
),
ranking_salary as (
select industry, avg_salary,
dense_rank() over (order by avg_salary desc) as
salary_rank
from avg_salary_by_industry
)
select * from ranking_salary where
salary_rank<=3
order by salary_rank;
```

[3] Find the top skill for each job industry. (Top skills are assumed to be based on # job postings that require those skills)

```
with industry_skill_count as (
	select ci.industry as company_industry,
jp.job_id, js.skill_abr
	from JobSkills js
	inner join JobPostings jp
	on js.job_id = jp.job_id
	inner join CompanyIndustries ci
	on jp.company_id = ci.company_id
	group by 1,2,3
),
skill_count_raw as (
select company_industry, skill_abr,
count(distinct job_id) as job_count
from industry_skill_count
group by 1,2
),
skill_ranking as (
select company_industry, skill_abr, job_count,
dense_rank() over (partition by
company_industry order by job_count desc) as
skill_rank
from skill_count_raw
)
select * from skill_ranking where skill_rank=1
order by company_industry, job_count desc;
```

[4] Which company has shown max employee count growth % over time?

```
with max_min_emp_count_raw as (
select company_id,
## Casting varchar to timestamp in case data
type is not already datetime ##
cast(time_recorded as datetime) as
date_time_stamp,
employee_count
from employee_counts
where time_recorded is not null
and coalesce(employee_count,0)>0
group by 1,2,3
),
min_max_rank as (
select company_id, date_time_stamp,
employee_count,
## Using row number since we want unique row
for a rank, dense rank may show duplicates ##
## Creating 2 ranks - 1 for earliest and 1 for
latest ##
row_number() over (partition by company_id
order by date_time_stamp) as min_rank,
row_number() over (partition by company_id
order by date_time_stamp desc) as max_rank
from  max_min_emp_count_raw
),
growth_calc as (
select coalesce(mn.company_id,
mx.company_id) as company_id,
# Including condition to make growth % as 0 if
denominator is 0, else do the actual percentage
change
case when
coalesce(mn.earliest_emp_count,0)>0 then
        (coalesce(mx.latest_emp_count,0) -
coalesce(mn.earliest_emp_count,0))*100/coale
sce(mn.earliest_emp_count,0)
  else 0 end as
emp_count_growth_change_percentage from
(select company_id, employee_count as
earliest_emp_count from min_max_rank where
min_rank=1) mn
left join
(select company_id, employee_count as
latest_emp_count from min_max_rank where
max_rank=1) mx
on mn.company_id = mx.company_id
)
select c.name as company_name, gc.* from
growth_calc gc inner join Companies c
on gc.company_id = c.company_id
order by
emp_count_growth_change_percentage desc;
```

[5] Which job industry has the highest number of openings? (Highest number of openings would mean the job_id doesn't have a closed time yet)

```
select * from
(
        select ci.industry as company_industry,
  count(distinct job_id) as jobs_count
        from JobPosting jp
        inner join CompanyIndustries ci
        on jp.company_id = ci.company_id
  where jp.closed_time_ts is not null
        group by 1
  order by jobs_count desc
) s
limit 1;
```

[6] Which companies have the highest number of specialties by industry?

```
with spec_count as (
        select industry, c.company_id, c.name
as company_name,
        count(distinct speciality) as
specialities_count
        from CompanySpecialities cs inner join
Companies c
        on cs.company_id = c.company_id
  inner join CompanyIndustries ci
  on ci.company_id = c.company_id
        group by 1,2,3
),
spec_rank as (
        select sc.*,
        dense_rank() over (partition by industry
order by specialities_count desc) as sp_rank
        from spec_count sc
)
select * from spec_rank
where sp_rank=1
order by specialities_count desc;
```

[7] What is the count of internships available with company name, size and job description?

```
select c.name as company_name,
c.company_size, jp.description as
job_description,
count(distinct job_id) as jobs_count
from JobPosting jp
inner join Companies c
on jp.company_id = c.company_id
where jp.closed_time is not null
and lower(trim(jp.work_type)) = 'internship'
group by 1,2,3
order by jobs_count desc;
```

[8] How many full time jobs offer medical, dental, and 401k as Benefits and have HQ in California?

```
# Filtering for all jobs with the requried benefits
with required_benefits_jobs as (
            select job_id
            from Benefits
            where lower(trim(type)) in ('401k',
'medical insurance', 'dental insurance')
            group by 1
)
select count(distinct jp.job_id) as jobs_count
from JobPosting jp inner join Companies c
on jp.company_id = c.company_id
inner join required_benefits_jobs ben
on jp.job_id = ben.job_id
where lower(trim(c.state)) in ('ca', 'california');
```

[9] Calculate average, maximum and minimum salaries for companies that are part of 'Nonprofit Organization Management' industry by each state

```
WITH ITCompanies AS (
   SELECT c.state, jp.max_salary, jp.min_salary
   FROM Companies c
   JOIN CompanyIndustries ci ON c.company_id
= ci.company_id
   JOIN JobPosting jp ON c.company_id =
jp.company_id
   WHERE ci.industry = 'Nonprofit Organization
Management'
)

SELECT state,
      max_salary AS max_salary,
      min_salary AS min_salary,
      AVG(max_salary) OVER (PARTITION BY
state) AS avg_max_salary,
      AVG(min_salary) OVER (PARTITION BY
state) AS avg_min_salary
FROM ITCompanies;
```

[10] List all companies in a specific city (ex: New York) and their average employee counts.

```
WITH CompanyEmployeeCounts AS (
   SELECT c.name, c.city, ec.employee_count,
         AVG(ec.employee_count) OVER
(PARTITION BY c.city) AS avg_employee_count
   FROM Companies c
   LEFT JOIN EmployeesCount ec ON
c.company_id = ec.company_id
)
SELECT name, city, avg_employee_count
FROM CompanyEmployeeCounts
WHERE city = 'New York';
```

[11] Find companies with a specific specialty (ex: Financial Services) that also have job postings with a certain skill (ex: ACCT) requirement:

```
WITH CompanyJobSkills AS (
   SELECT c.name as company_name,
cs.speciality, js.skill_abr,
         ROW_NUMBER() OVER (PARTITION BY
c.name, cs.speciality, js.skill_abr) AS rn
   FROM Companies c
   JOIN CompanySpecialities cs ON
c.company_id = cs.company_id
   JOIN JobPosting jp ON c.company_id =
jp.company_id
   JOIN JobSkills js ON jp.job_id = js.job_id
)
SELECT 'Job Roll', speciality, skill_abr
FROM CompanyJobSkills
WHERE speciality = 'Financial Services' AND
skill_abr = 'ACCT'
 AND rn = 1
ORDER BY company_name;
```

## STORED PROCEDURE

### [1] GetSkillsAbrByCompany

```
DELIMITER //
```

```
CREATE PROCEDURE GetSkillsAbrByCompany(IN
companyID_input BIGINT)
BEGIN
  SELECT name as company_name, skill_abr,
COUNT(js.job_id) AS job_count
  FROM JobSkills js JOIN JobPosting jp ON
js.job_id =jp.job_id
  JOIN Companies c ON jp.company_id =
c.company_id
  WHERE c.company_id = companyID_input
  GROUP BY skill_abr, company_name
  ORDER BY job_count DESC;
END //
DELIMITER ;
```

Example of usage: call
GetSkillsAbrByCompany(1016);

[2] UpdateJobPostingSalary (Updating new salary to max salary)

```
DELIMITER $$

CREATE PROCEDURE
UpdateJobPostingSalary1(IN p_job_ID BIGINT,
newSalary DECIMAL(10,2))
BEGIN
	UPDATE JobPosting
   SET max_salary = newSalary
   WHERE job_id = p_job_ID;
END $$

DELIMITER ;

SET SQL_SAFE_UPDATES = 0;
```

Example of usage: call
UpdateJobPostingSalary1(133114754,
80000.00);

[3] GetJobPostingsByLocation

```
DELIMITER //
CREATE PROCEDURE
GetJobPostingsByLocation(IN joblocation
VARCHAR(200))
BEGIN
   SET @joblocation = joblocation;
   SELECT job_id, title FROM JobPosting WHERE
location = @joblocation;
END //
DELIMITER ;
```

Example of usage: CALL
GetJobPostingsByLocation('New York NY');

## TRIGGERS

[1] BeforeJobPostingUpdate

```
DELIMITER //
CREATE TRIGGER BeforeJobPostingUpdate
BEFORE UPDATE ON Jobposting
FOR EACH ROW
BEGIN
  IF NEW.expiry < OLD.expiry THEN
    SET NEW.closed_time = NOW();
  END IF;
END;
//
DELIMITER ;
```

[2] AfterJobPostingInsert

```
DELIMITER //
CREATE TRIGGER AfterJobPostingInsert
AFTER INSERT ON Jobposting
FOR EACH ROW
BEGIN
  INSERT INTO JobPostingLog (job_id,
company_id, title, inserted_at)
  VALUES (NEW.job_id, NEW.company_id,
NEW.title, NOW());
END;
//
DELIMITER ;
```

[3] AfterSkillsUpdate

```
DELIMITER //
CREATE TRIGGER AfterSkillsUpdate
AFTER UPDATE ON Jobskills
FOR EACH ROW
BEGIN
  INSERT INTO NotificationLog (message,
recipient, sent_at)
  VALUES ('Skills for job ' or NEW.job_id or ' have
been updated.', 'HR Department', NOW());
END;
//
DELIMITER ;
```

[4] AfterSkillsForRemoteWorkingInsert

```
DELIMITER //
CREATE TRIGGER
AfterSkillsForRemoteWorkingInsert
AFTER INSERT ON Jobskills
FOR EACH ROW
BEGIN
  IF NEW.skill_abr = 'RemoteWork' THEN
    INSERT INTO NotificationLog (message,
recipient, sent_at)
    VALUES ('Remote work skills added for job '
OR NEW.job_id, 'Remote Work Department',
NOW());
  END IF;
END;
//
DELIMITER ;
```

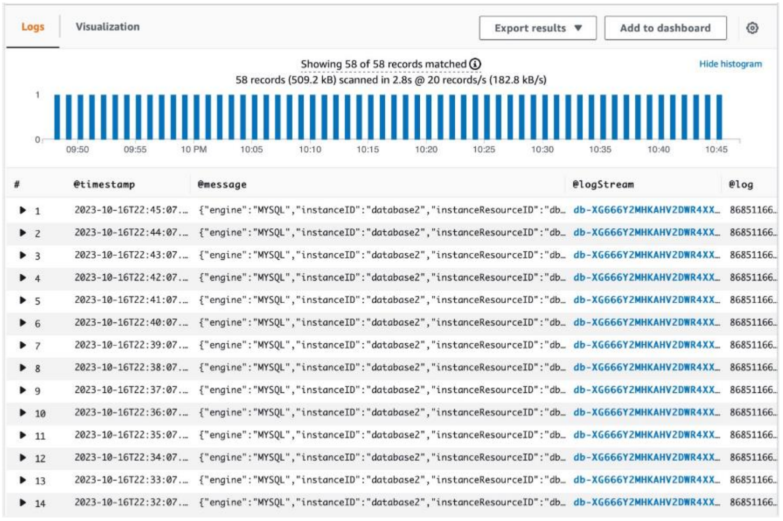
## ACCESS PRIVILEGES

ACCESS PRIVILEGES:
USER CREATIONS FOR ACCESS PRIVILEGES

```
GRANT SELECT, CREATE VIEW ,SHOW VIEW
Host,User FROM mysql.user;
CREATE USER     '<DB_USER>'
IDENTIFIED BY '<DBLAB@1>';
```

```
OR GRANT SELECT, CREATE VIEW, SHOW VIEW
ON  linkedinjobposting_lab1.* TO '<DB_USER>';
CREATE USER '<DB_USER>' IDENTIFIED BY
'<DBLAB@1>';
```

## LOGGING OF DATABASE:



Showing 58 of 58 records matched
58 records (509.2 kB) scanned in 2.8s @ 20 records/s (182.8 kB/s)

| # | @timestamp | @message | @logStream | @log |
|---|---|---|---|---|
| 1 | 2023-10-16T22:45:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 2 | 2023-10-16T22:44:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 3 | 2023-10-16T22:43:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 4 | 2023-10-16T22:42:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 5 | 2023-10-16T22:41:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 6 | 2023-10-16T22:40:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 7 | 2023-10-16T22:39:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 8 | 2023-10-16T22:38:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 9 | 2023-10-16T22:37:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 10 | 2023-10-16T22:36:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 11 | 2023-10-16T22:35:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 12 | 2023-10-16T22:34:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 13 | 2023-10-16T22:33:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |
| 14 | 2023-10-16T22:32:07.… | {"engine":"MYSQL","instanceID":"database2","instanceResourceID":"db… | db-XG666Y2MHKAHV2DWR4XX… | 86851166… |

## AWS CONNECTIVITY WITH PYTHON:

```
import mysql.connector

# RDS instance details
db_endpoint = 'database2.cnunqj562ud9.us-
east-1.rds.amazonaws.com'
db_port = 3306
db_user = 'admin'
db_password = 'admin123'
db_name = 'linkedinjobposting_lab1'

try:
  # Connect to the RDS database
  connection = mysql.connector.connect(
    host=db_endpoint,
    port=db_port,
    user=db_user,
    password=db_password,
        database=db_name
    )

    if connection.is_connected():
        print(f"Connected to {db_endpoint} on port
{db_port} as user {db_user} to database
{db_name}")

        # Create a cursor for database operations
        cursor = connection.cursor()

        # Execute Procedure
        print("create stored procedure for skills by
company and call the procedure")
        add_triggers(procedure_sql)
        cursor.callproc(procedure_name,[1016])
        print("create stored procedure for
Updating new salary to max salary and call the
procedure")
        add_triggers(procedure_sql1)

cursor.callproc(procedure_name1,[133114754,
80000.00])
        # Execute Triggers
        print("This trigger creates a notification in
the NotificationLog table whenever skills
associated with a job are updated in the
Jobskills table.")

add_triggers(NOTIFICATION_LOG_TRIGGER)
        print("This trigger creates a notification in
the NotificationLog table whenever a new skill
is inserted into the Jobskills table.")
        add_triggers(NEW_SKILL_INSERT_TRIGGER)
        print("This trigger creates a log entry in the
JobPostingLog table whenever a new job
posting is inserted into the Jobposting table.")
        add_triggers(JOB_POSTING_LOG_TRIGGER)
        print("This trigger indicates that the job
posting is now closed.")

add_triggers(CLOSED_JOB_POSTING_TRIGGER)
        # Define your SQL SELECT statement
        print ("Calculate average maximum and
minimum salaries for Nonprofit Organization
Management companies while preserving the
details of each company within the state")
        print_query_data(MAXMIN_SALARY)
        print ("List all companies in a specific city
and their average employee counts.")

print_query_data(COMPANIES_SPECIFIC_CITY)
        print ("Find companies job roll with a
specific speciality that also have job postings
with a certain skill requirement:")

print_query_data(JOBROLL_SPECIFIC_SPECIALIT
Y)
        print("Average # benefits per job and
details of jobs offering higher than the average
# of benefits")
        print_query_data(FIRST_CTE_BENEFITS)
        print ("Find the average salary by each job
industry. Rank to find the top 3 industries by
avg salary (joins and window function)")

print_query_data(AVERAGE_SALARY_BY_EACH_
INDUSTRY)
        print("Find the top skill for each job
industry")

print_query_data(TOP_SKILL_BY_EACH_INDUST
RY)
        print ("Which company has shown max
employee count growth % over time?
(headcount at latest ts - head count at earliest
ts)/head count at earliest ts deduping and
excluding null values")

print_query_data(MAX_EMPLOYEE_GROWTH_
COUNT)
        print(" Which job industry has the highest
number of openings, Highest number of
openings would mean the job_id that don't
have a closed time yet")

print_query_data(JOB_OPENINGS_WITH_HIGHE
ST_NUMBER_OF_OPENINGS)
        print ("How many full time jobs offer
medical, dental, and 401k as benefits anda have
HQ in California? Filtering for all jobs with the
requried benefits")
        print_query_data(FULL_TIME_JOBS)
        print ("Which companies have the highest
number of specialties by industry?")
```

```
print_query_data(HIGHEST_NUMBER_OF_SPECI
ALITIES)
        print("What is the count of internships
available with company name, size and job
description?")

print_query_data(COUNT_OF_INTERNSHIPS)

except mysql.connector.Error as err:
    print(f"Error: {err}")
finally:
    # Close the cursor and connection
    if 'cursor' in locals():
        cursor.close()
    if 'connection' in locals():
        connection.close()
        print("Database connection closed.")
```

## PERFORMANCE MEASUREMENT:

### *AWS*

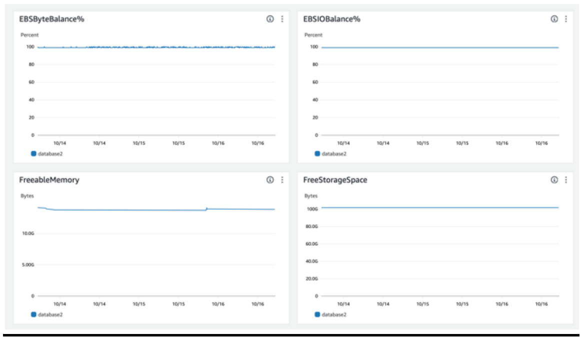


*Figure 2 AWS Performance Screenshot*

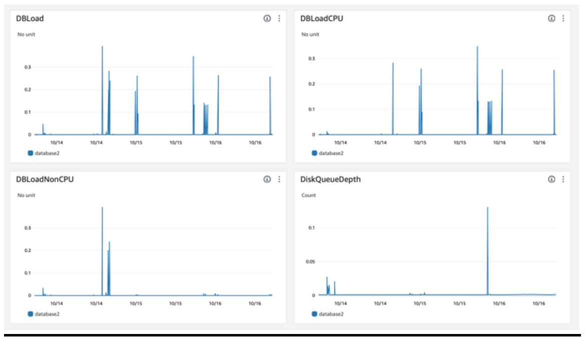


*Figure 3 AWS Performance*

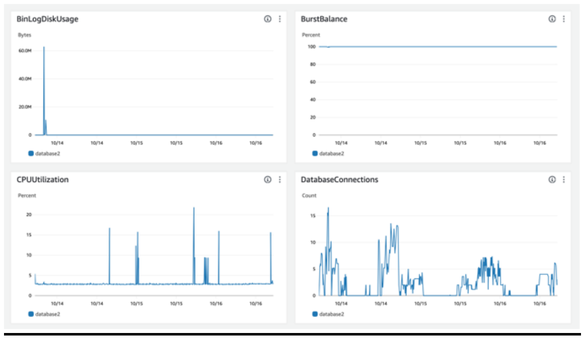


*Figure 4 AWS Performance*