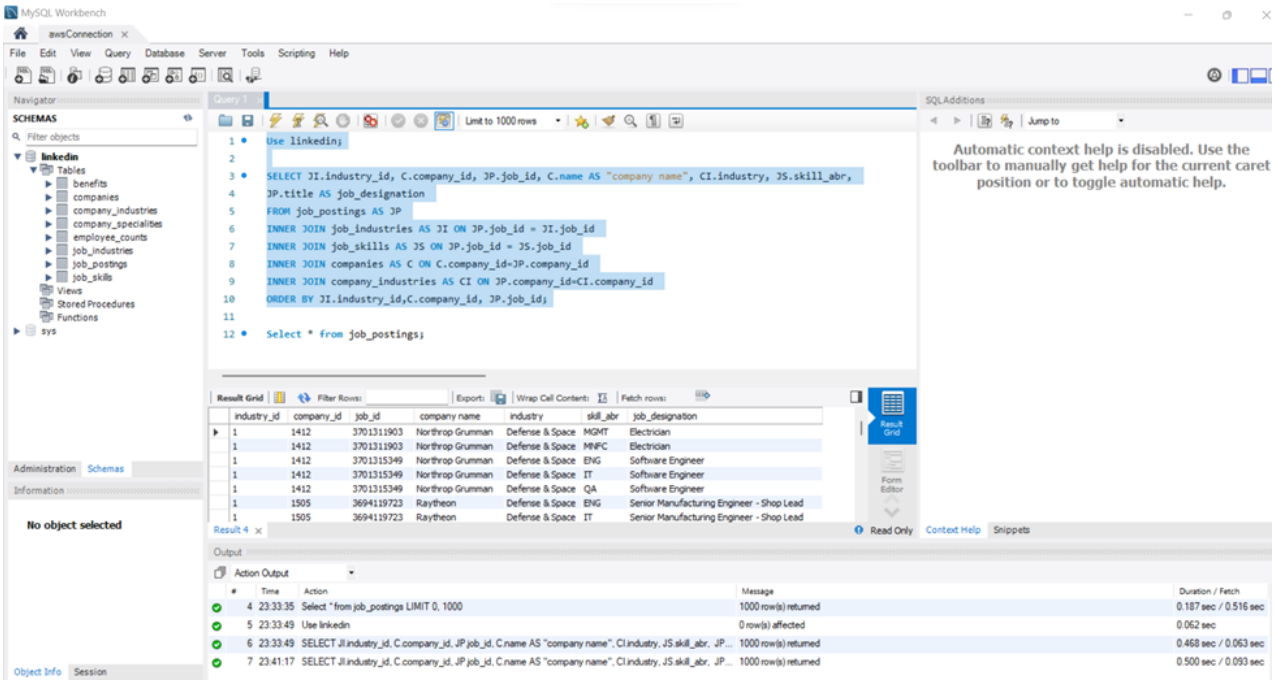
Functional Analysis

1.Job Categorization based on industries, companies and job functions.

```
Use linkedin;

SELECT JI.industry_id, C.company_id, JP.job_id, C.name AS "company name", CI.industry, JS.skill_abr,

JP.title AS job_designation

FROM job_postings AS JP

INNER JOIN job_industries AS JI ON JP.job_id = JI.job_id

INNER JOIN job_skills AS JS ON JP.job_id = JS.job_id

INNER JOIN companies AS C ON C.company_id=JP.company_id

INNER JOIN company_industries AS CI ON JP.company_id=CI.company_id

ORDER BY JI.industry_id,C.company_id, JP.job_id;
```



2.Geographical distribution of Benefits and job roles provided by companies.

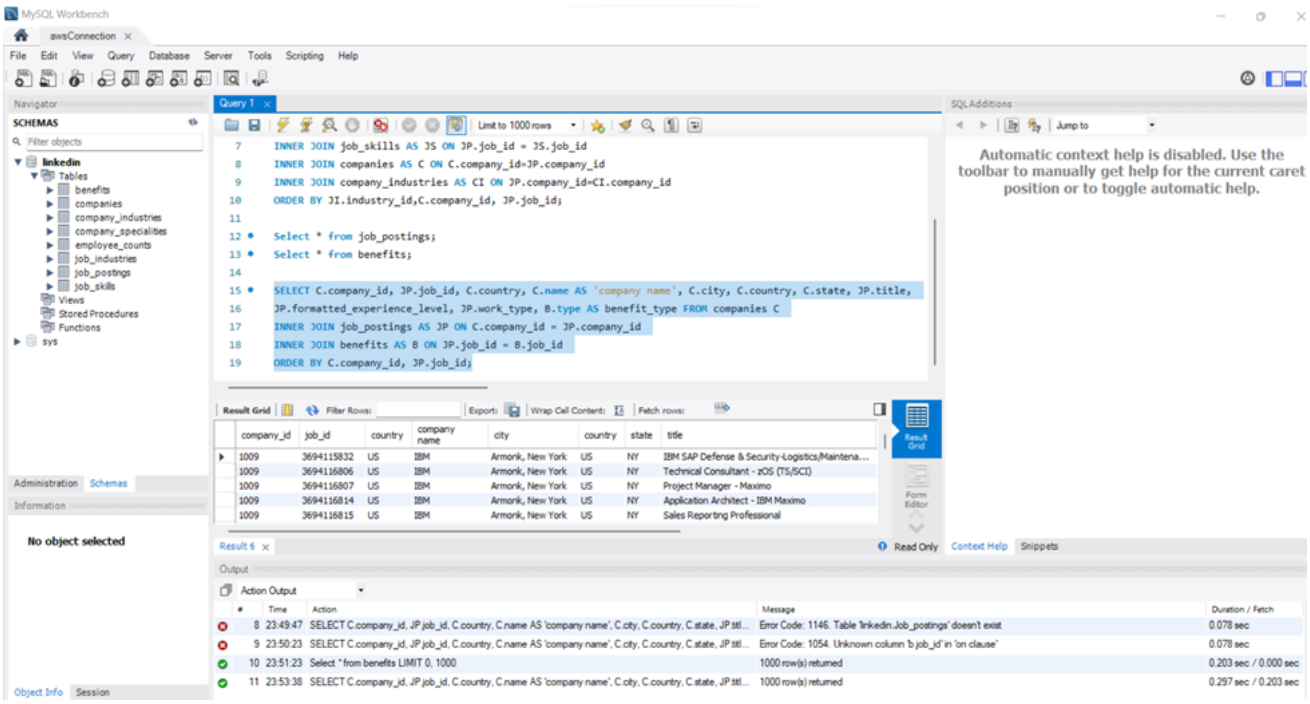
SELECT C.company_id, JP.job_id, C.country, C.name AS 'company name', C.city, C.country, C.state, JP.title,

JP.formatted_experience_level, JP.work_type, B.type AS benefit_type FROM companies C

INNER JOIN job_postings AS JP ON C.company_id = JP.company_id

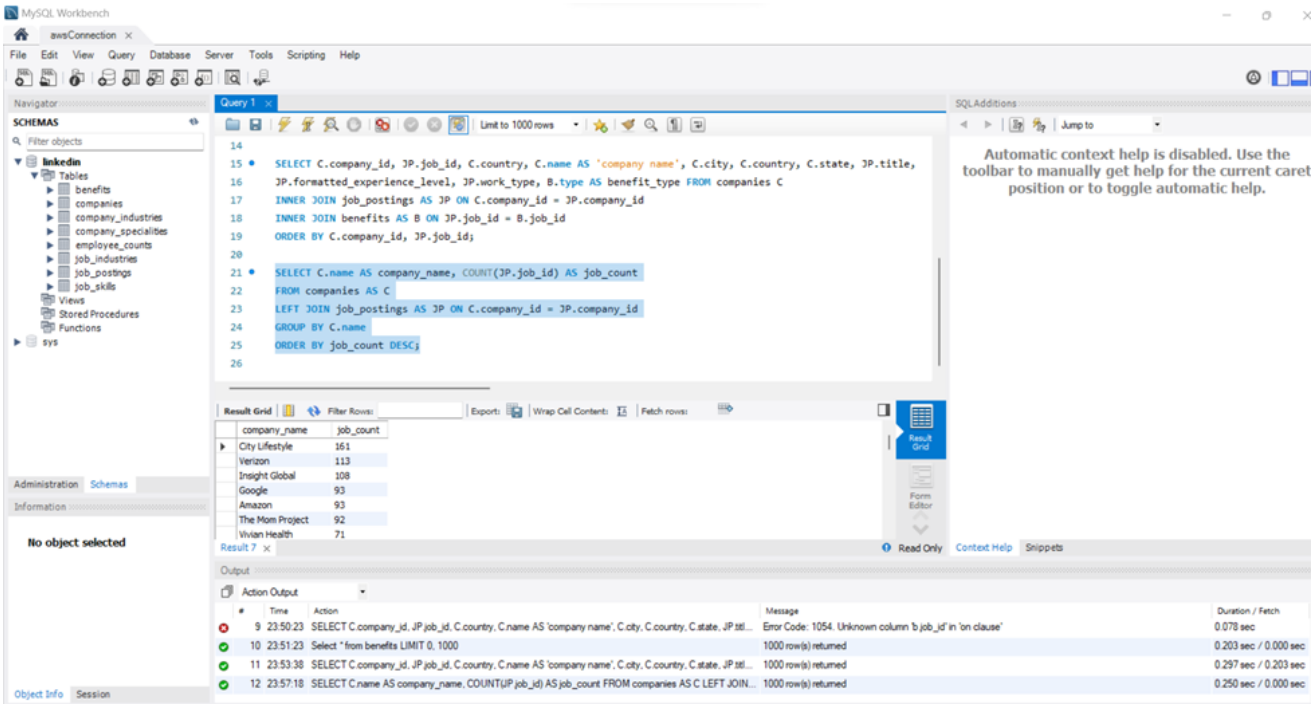
INNER JOIN benefits AS B ON JP.job_id = B.job_id

ORDER BY C.company_id, JP.job_id;



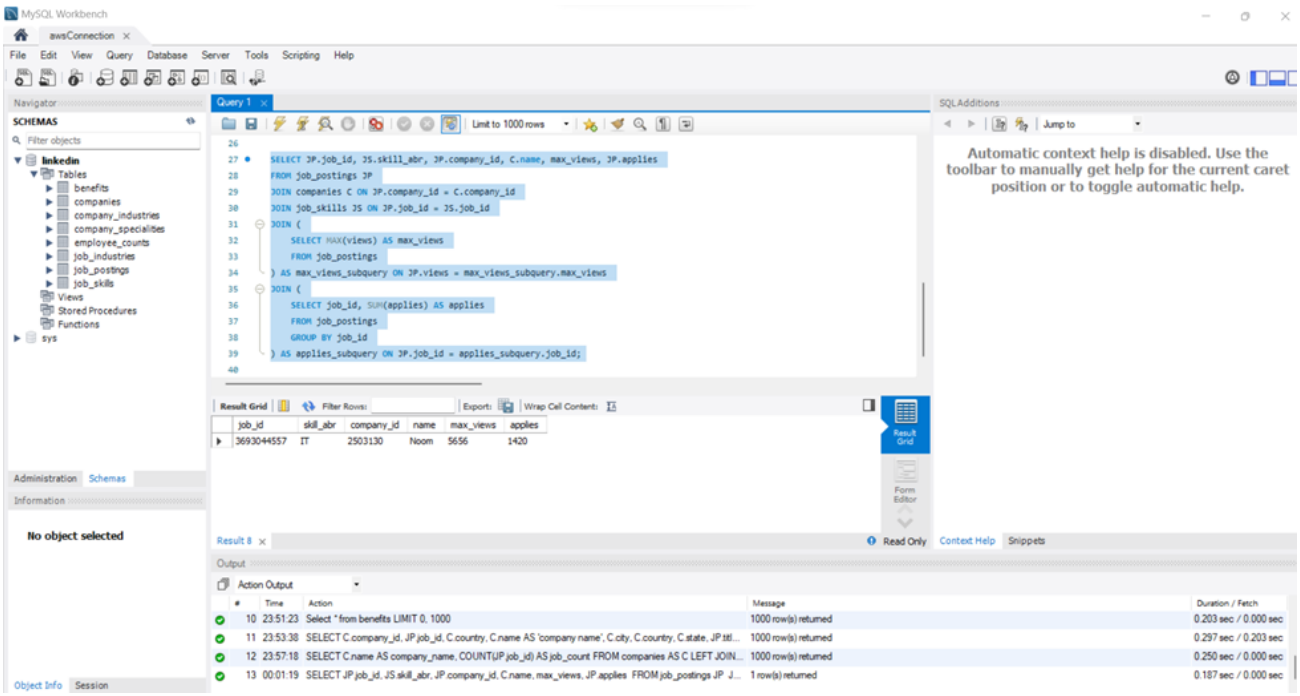
3.Total job postings listed by each company.

```
SELECT C.name AS company_name, COUNT(JP.job_id) AS job_count
FROM companies AS C
LEFT JOIN job_postings AS JP ON C.company_id = JP.company_id
GROUP BY C.name
ORDER BY job_count DESC;
```



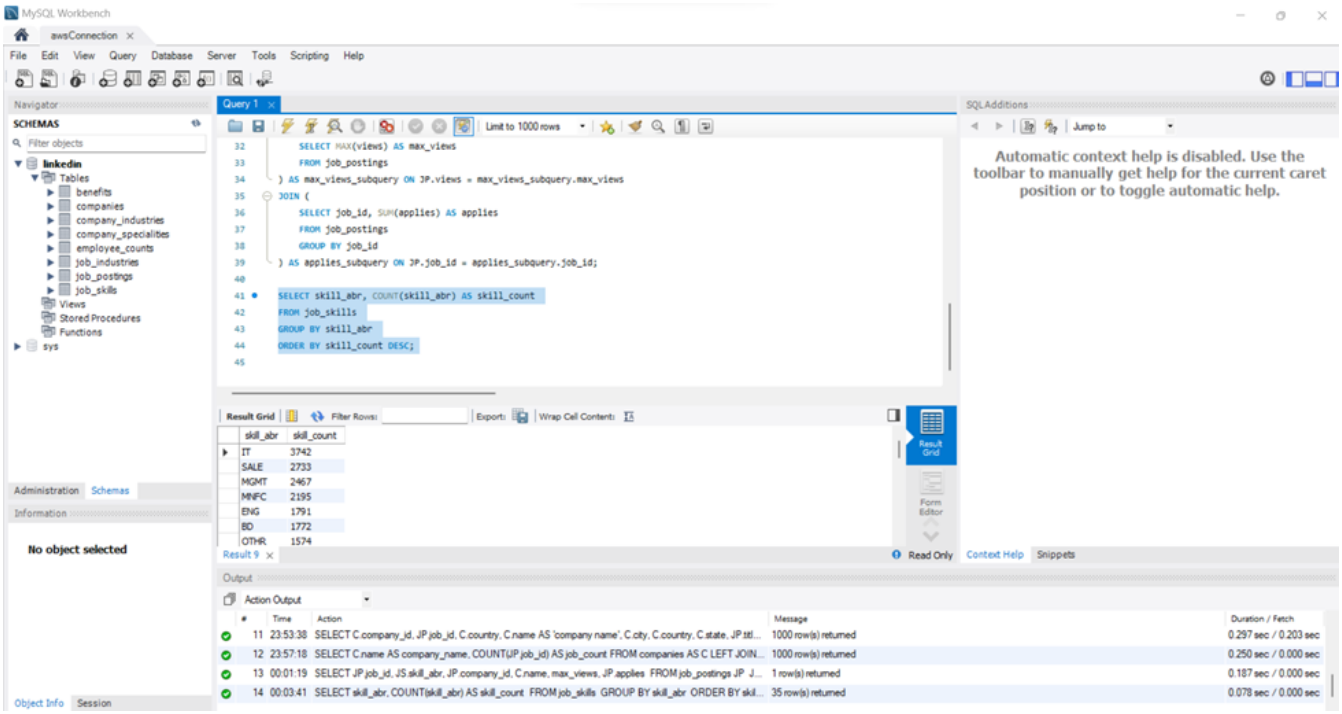
4. **Job posting with maximum no. of views and total number of applications**.

```
SELECT JP.job_id, JS.skill_abr, JP.company_id, C.name, max_views, JP.applies
FROM job_postings JP
JOIN companies C ON JP.company_id = C.company_id
JOIN job_skills JS ON JP.job_id = JS.job_id
JOIN (
   SELECT MAX(views) AS max_views
   FROM job_postings
) AS max_views_subquery ON JP.views = max_views_subquery.max_views
JOIN (
   SELECT job_id, SUM(applies) AS applies
   FROM job_postings
   GROUP BY job_id
) AS applies_subquery ON JP.job_id = applies_subquery.job_id;
```


MySQL Workbench
awsConnection
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
linkedin
Tables
benefits
companies
company_industries
company_specialities
employee_counts
job_industries
job_postings
job_skills
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
SELECT JP.job_id, JS.skill_abr, JP.company_id, C.name, max_views, JP.applies
FROM job_postings JP
JOIN companies C ON JP.company_id = C.company_id
JOIN job_skills JS ON JP.job_id = JS.job_id
JOIN (
SELECT MAX(views) AS max_views
FROM job_postings
) AS max_views_subquery ON JP.views = max_views_subquery.max_views
JOIN (
SELECT job_id, SUM(applies) AS applies
FROM job_postings
GROUP BY job_id
) AS applies_subquery ON JP.job_id = applies_subquery.job_id;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
job_id skill_abr company_id name max_views applies
3693044557 IT 2503130 Noom 5656 1420
Result Grid
Form Editor
Read Only
Result 8
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
10 23:51:23 Select * from benefits LIMIT 0, 1000 1000 row(s) returned 0.203 sec / 0.000 sec
11 23:53:38 SELECT C.company_id, JP.job_id, C.country, C.name AS 'company name', C.city, C.country, C.state, JP.titl... 1000 row(s) returned 0.297 sec / 0.203 sec
12 23:57:18 SELECT C.name AS company_name, COUNT(JP.job_id) AS job_count FROM companies AS C LEFT JOIN... 1000 row(s) returned 0.250 sec / 0.000 sec
13 00:01:19 SELECT JP.job_id, JS.skill_abr, JP.company_id, C.name, max_views, JP.applies FROM job_postings JP J... 1 row(s) returned 0.187 sec / 0.000 sec

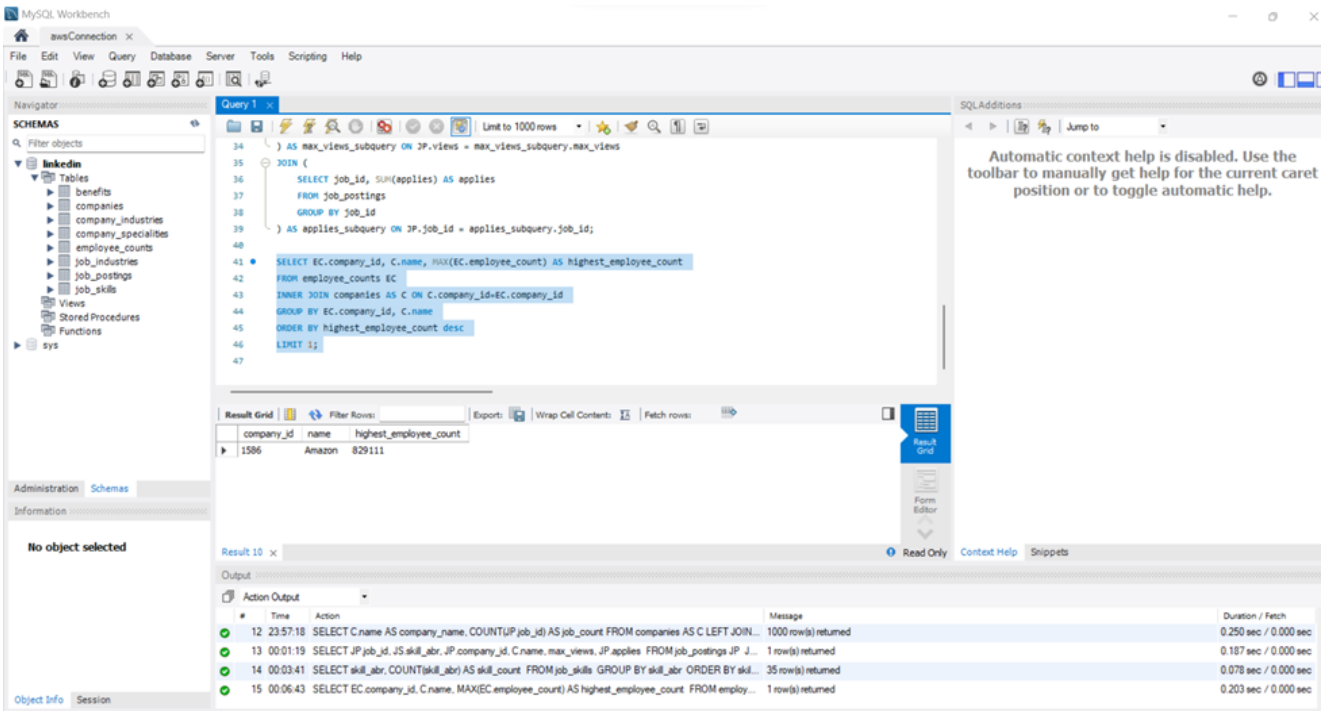
5.Identifying skills in demand.

```
SELECT skill_abr, COUNT(skill_abr) AS skill_count
FROM job_skills
GROUP BY skill_abr
ORDER BY skill_count DESC;
```



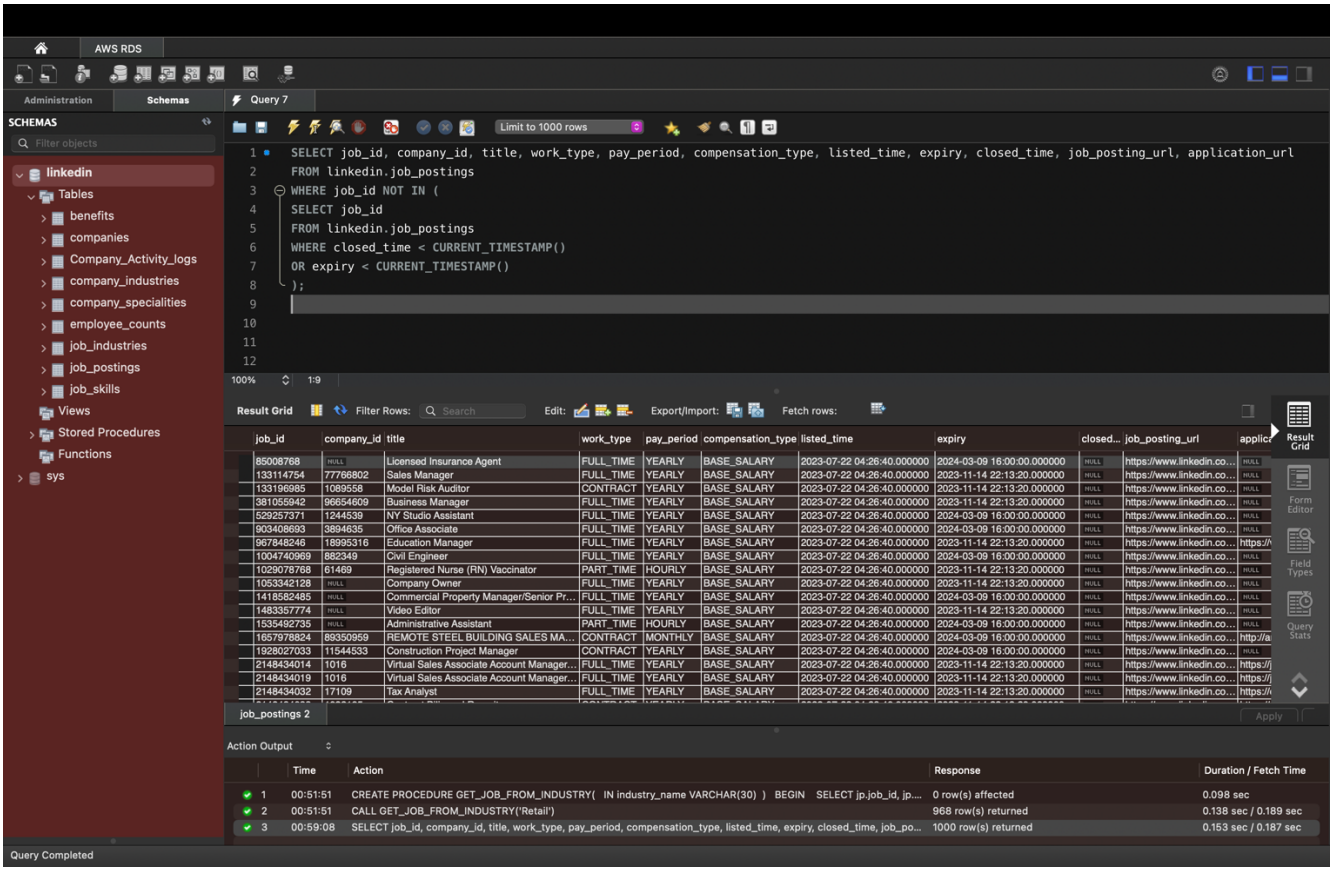
6. **Company having highest employee count.**

```
SELECT EC.company_id, C.name, MAX(EC.employee_count) AS highest_employee_count
FROM employee_counts EC
INNER JOIN companies AS C ON C.company_id=EC.company_id
GROUP BY EC.company_id, C.name
ORDER BY highest_employee_count desc
LIMIT 1;
```


MySQL Workbench
awsConnection
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
linkedin
Tables
benefits
companies
company_industries
company_specialities
employee_counts
job_industries
job_postings
job_skills
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
No object selected
Object Info Session
Query 1
Limit to 1000 rows
) AS max_views_subquery ON JP.views = max_views_subquery.max_views
JOIN (
SELECT job_id, SUM(applies) AS applies
FROM job_postings
GROUP BY job_id
) AS applies_subquery ON JP.job_id = applies_subquery.job_id;
SELECT EC.company_id, C.name, MAX(EC.employee_count) AS highest_employee_count
FROM employee_counts EC
INNER JOIN companies AS C ON C.company_id=EC.company_id
GROUP BY EC.company_id, C.name
ORDER BY highest_employee_count desc
LIMIT 1;
Result Grid
Filter Rows:
Export:
Wrap Cell Contents:
Fetch rows:
company_id name highest_employee_count
1586 Amazon 829111
Result Grid
Form Editor
Result 10
Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
12 23:57:18 SELECT C.name AS company_name, COUNT(JP.job_id) AS job_count FROM companies AS C LEFT JOIN... 1000 row(s) returned 0.250 sec / 0.000 sec
13 00:01:19 SELECT JP.job_id, JS.skill_abr, JP.company_id, C.name, max_views, JP.applies FROM job_postings JP J... 1 row(s) returned 0.187 sec / 0.000 sec
14 00:03:41 SELECT skill_abr, COUNT(skill_abr) AS skill_count FROM job_skills GROUP BY skill_abr ORDER BY skill... 35 row(s) returned 0.078 sec / 0.000 sec
15 00:06:43 SELECT EC.company_id, C.name, MAX(EC.employee_count) AS highest_employee_count FROM employ... 1 row(s) returned 0.203 sec / 0.000 sec

7. Job postings that are currently active, not expired or closed.

```
SELECT job_id, company_id, title, work_type, pay_period, compensation_type, listed_time, expiry,
closed_time, job_posting_url, application_url
FROM linkedin.job_postings
WHERE job_id NOT IN (
SELECT job_id
FROM linkedin.job_postings
WHERE closed_time < CURRENT_TIMESTAMP()
OR expiry < CURRENT_TIMESTAMP()
);
```



8.**Location wise top 10 companies with the most job postings**

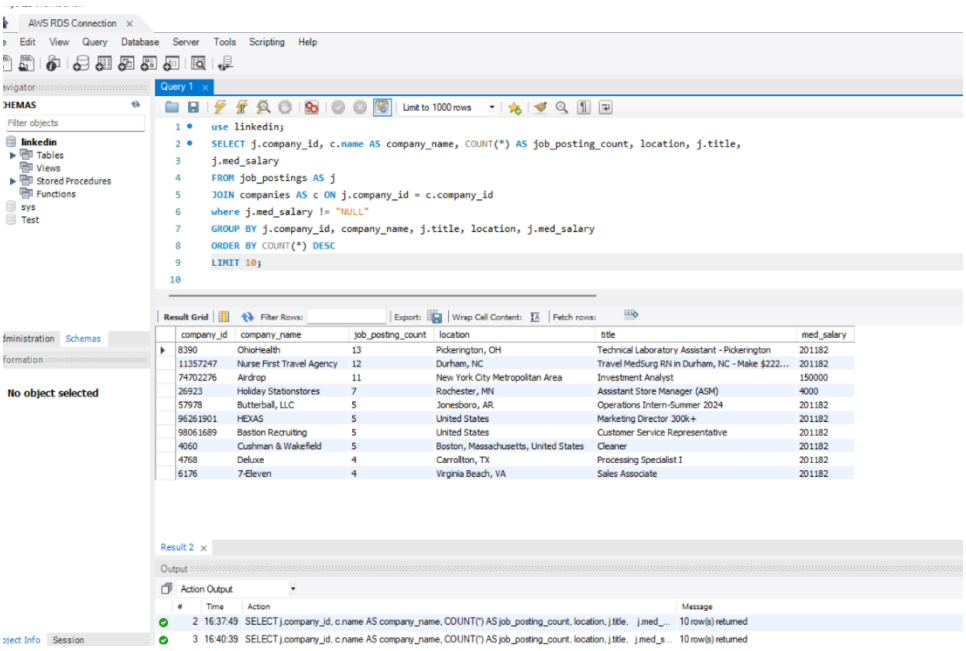
```
SELECT j.company_id, c.name AS company_name, COUNT(*) AS job_posting_count, location, j.title,
j.med_salary
FROM job_postings AS j
JOIN companies AS c ON j.company_id = c.company_id
```

where j.med_salary != "NULL"

GROUP BY j.company_id, company_name, j.title, location, j.med_salary

ORDER BY COUNT(*) DESC

LIMIT 10;



9. **Analysis of Job Postings with on Location, Company, Title, and Application Duration and filtered through Inferred through text by linkedin**

SELECT j.job_id,j.location, c.name, j.title, datediff(expiry,listed_time) as Days_to_apply

FROM job_postings_old AS j

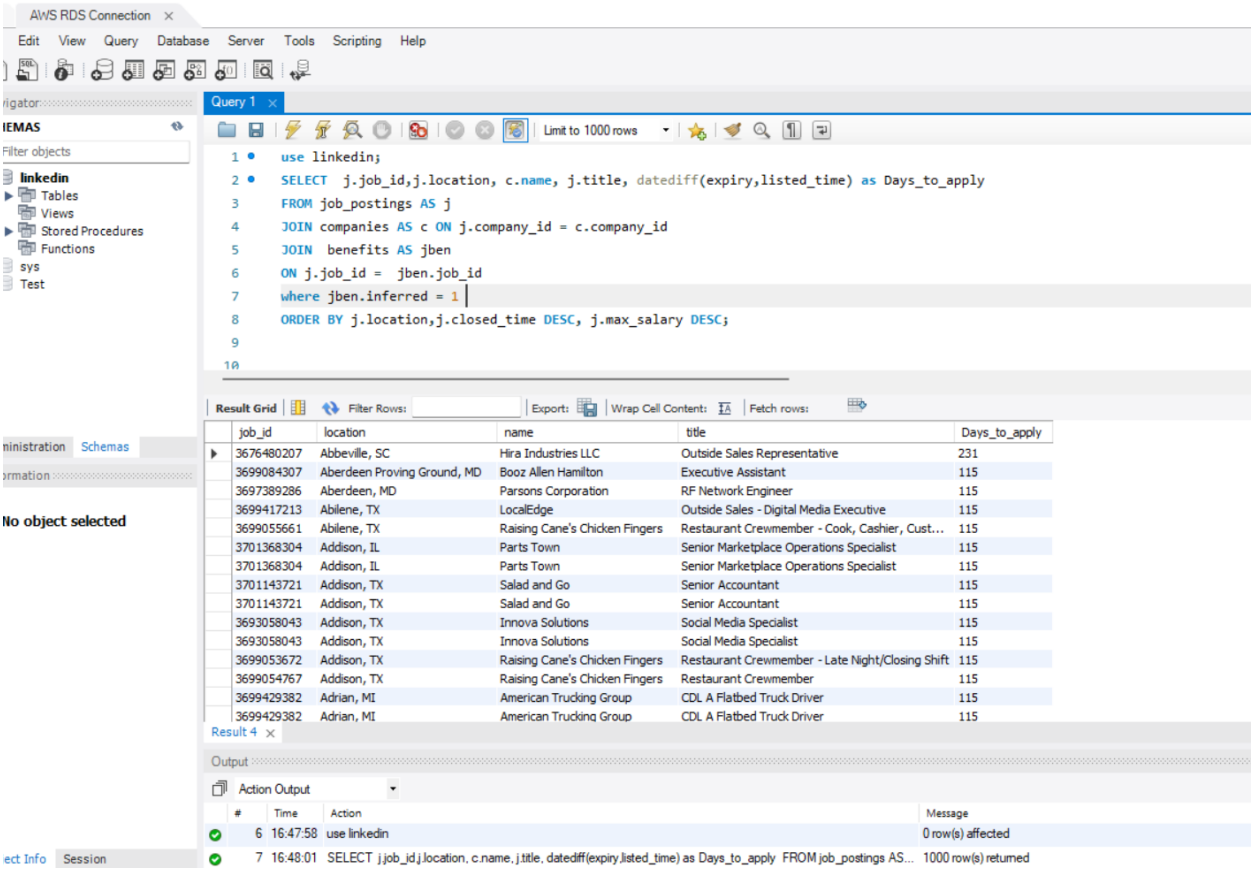
JOIN companies AS c ON j.company_id = c.company_id

JOIN benefits AS jben

ON j.job_id = jben.job_id

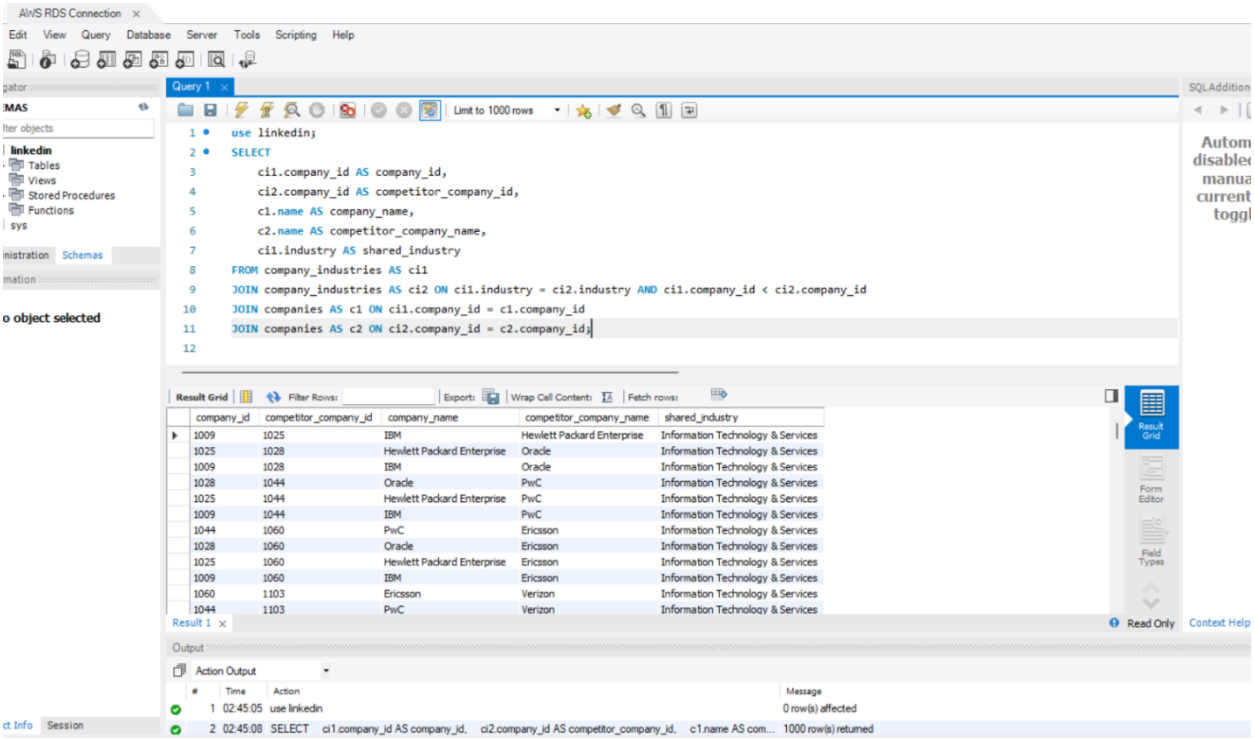
where jben.inferred = 1

ORDER BY j.location,j.closed_time DESC, j.max_salary DESC;



10. **Companies and their competitors sharing the same industry**

```
SELECT
ci1.company_id AS company_id,
ci2.company_id AS competitor_company_id,
c1.name AS company_name,
c2.name AS competitor_company_name,
ci1.industry AS shared_industry
FROM company_industries AS ci1
JOIN company_industries AS ci2 ON ci1.industry = ci2.industry AND ci1.company_id < ci2.company_id
JOIN companies AS c1 ON ci1.company_id = c1.company_id
JOIN companies AS c2 ON ci2.company_id = c2.company_id;
```



11. Analyze Job Demand Over month:

SELECT

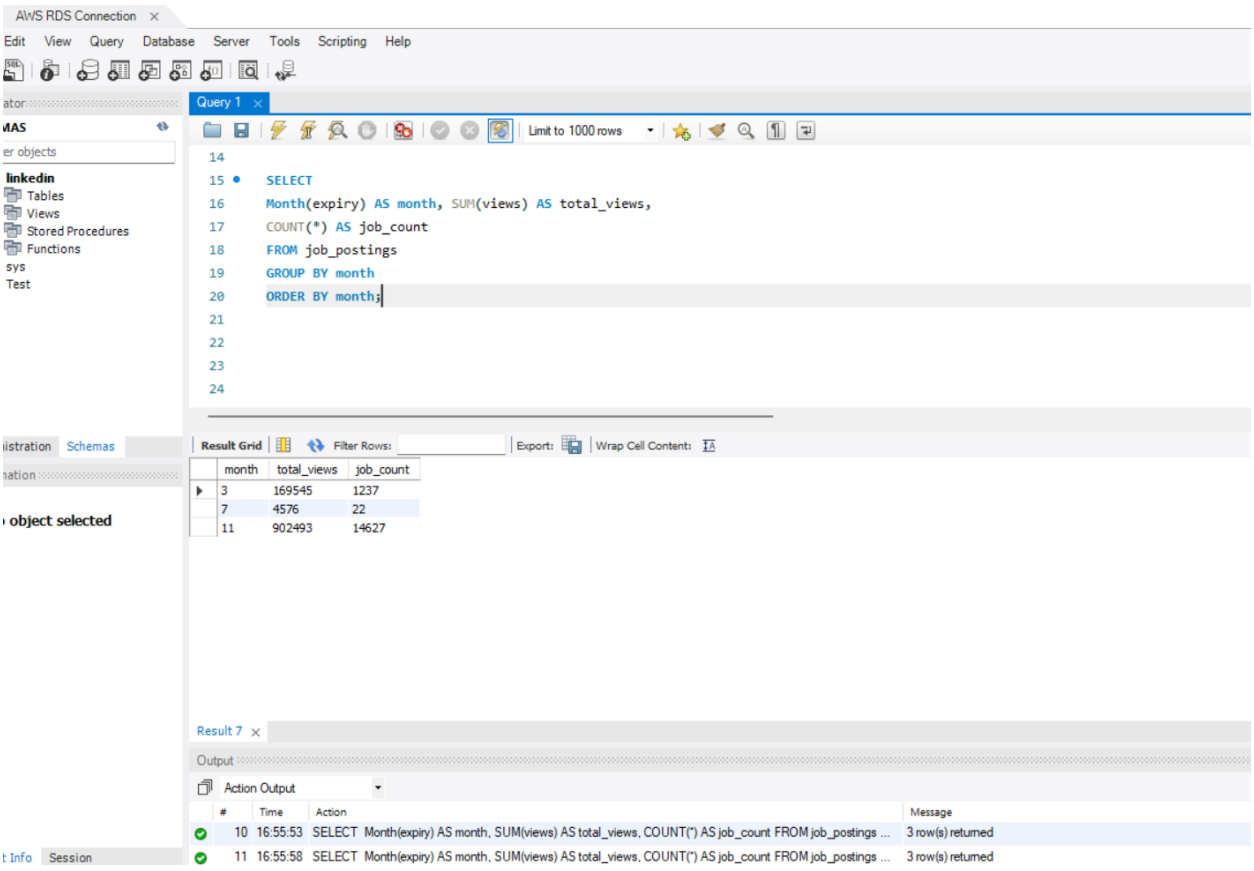
Month(expiry) AS month, SUM(views) AS total_views,

COUNT(*) AS job_count

FROM job_postings_old

GROUP BY month

ORDER BY month;



12. Partition Method to get the Annual salary on the max_salary specifying range in where clause.

```
SELECT

j.company_id,

j.formatted_work_type,

j.pay_period,

j.max_salary AS max_salary,

max(round(j.max_salary * 12, 3)) OVER (PARTITION BY j.company_id) AS Annual_Salary

FROM job_postings j

where (company_id BETWEEN 1500 AND 2500) and pay_period ='MONTHLY';
```

13. Analyzing the count of different work types such as Full-Time, Part-Time and Contractor Roles based on their pay_period

SELECT

pay_period,

sum(formatted_work_type = 'Full-time') AS "Full-Time Roles",

sum(formatted_work_type = 'Part-time') AS "Part-Time Roles",

sum(formatted_work_type = 'Contract') AS "Contract Roles"

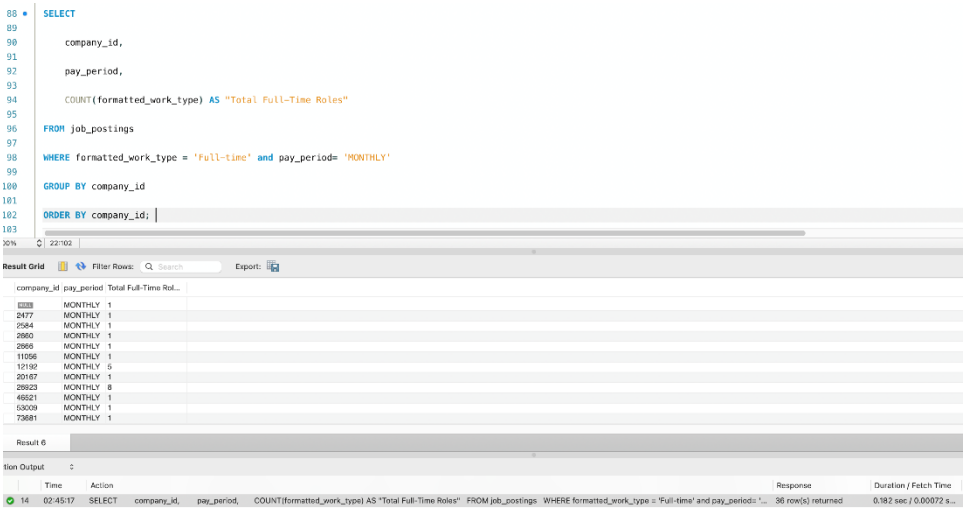
FROM job_postings

GROUP BY pay_period

ORDER BY pay_period;

14. Analyzing the number of Full-Time opportunities offered for Monthly pay_period by each company

```
SELECT

company_id,

pay_period,

COUNT(formatted_work_type) AS "Total Full-Time Roles"

FROM job_postings

WHERE formatted_work_type = 'Full-time' and pay_period= 'MONTHLY'

GROUP BY company_id

ORDER BY company_id;
```



15. Display the Total Roles in each roles types by every company.

```
SELECT

company_id,

formatted_work_type,

pay_period,
```

COUNT(*) OVER (PARTITION BY company_id ORDER BY pay_period) AS "Total Roles"

FROM job_postings

WHERE pay_period != 'NA' and company_id != 0

ORDER BY pay_period;

16. Partition Method to get the Annual salary on the max_salary specifying range in where clause.

SELECT

j.company_id,

j.formatted_work_type,

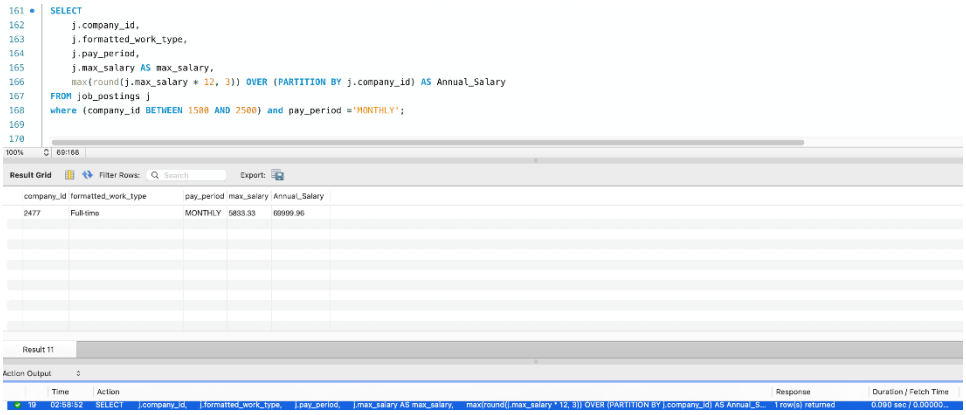
j.pay_period,

j.max_salary AS max_salary,

max(round(j.max_salary * 12, 3)) OVER (PARTITION BY j.company_id) AS Annual_Salary

FROM job_postings j

where (company_id BETWEEN 1500 AND 2500) and pay_period ='MONTHLY';

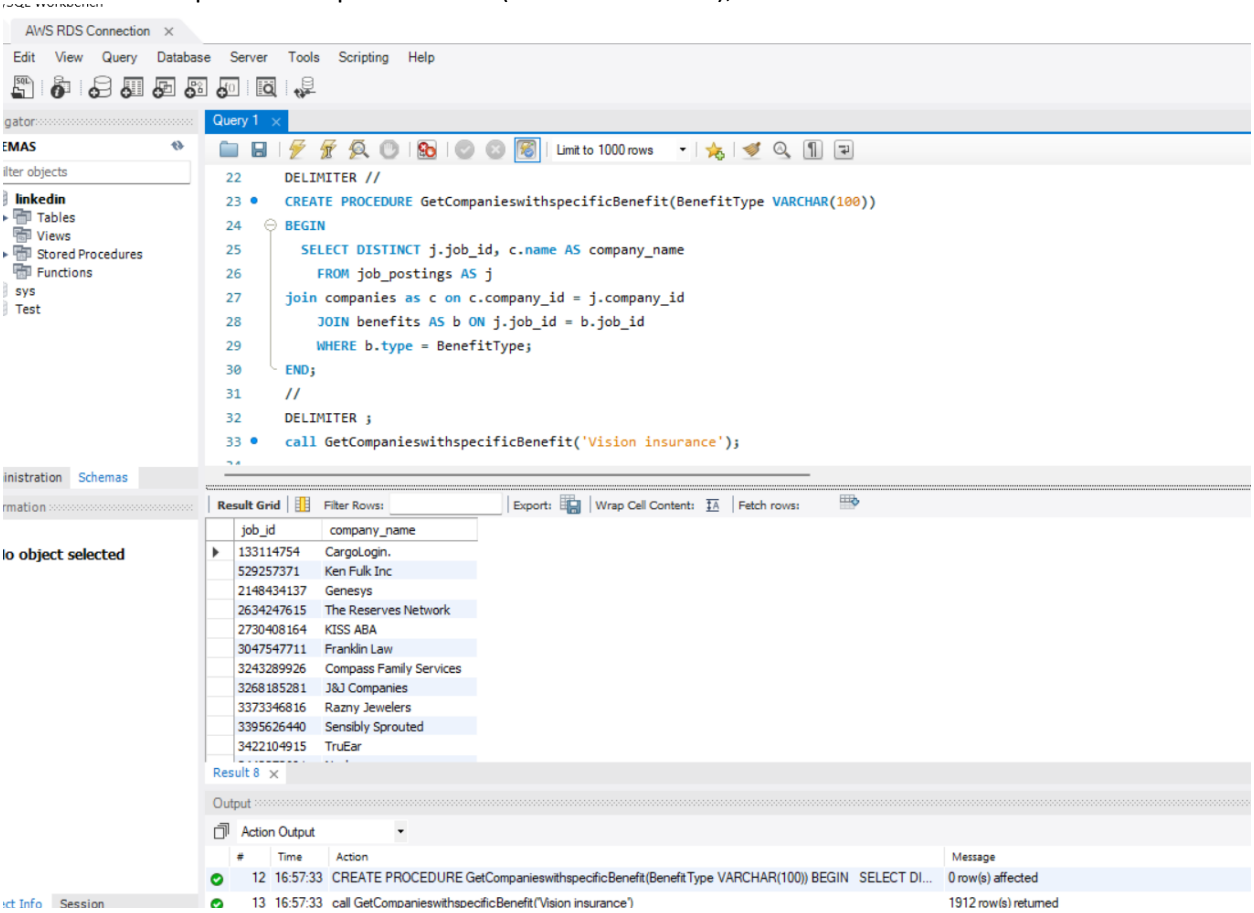


STORED PROCEDURES:

1. **Companies with vision Insurance as benefit**

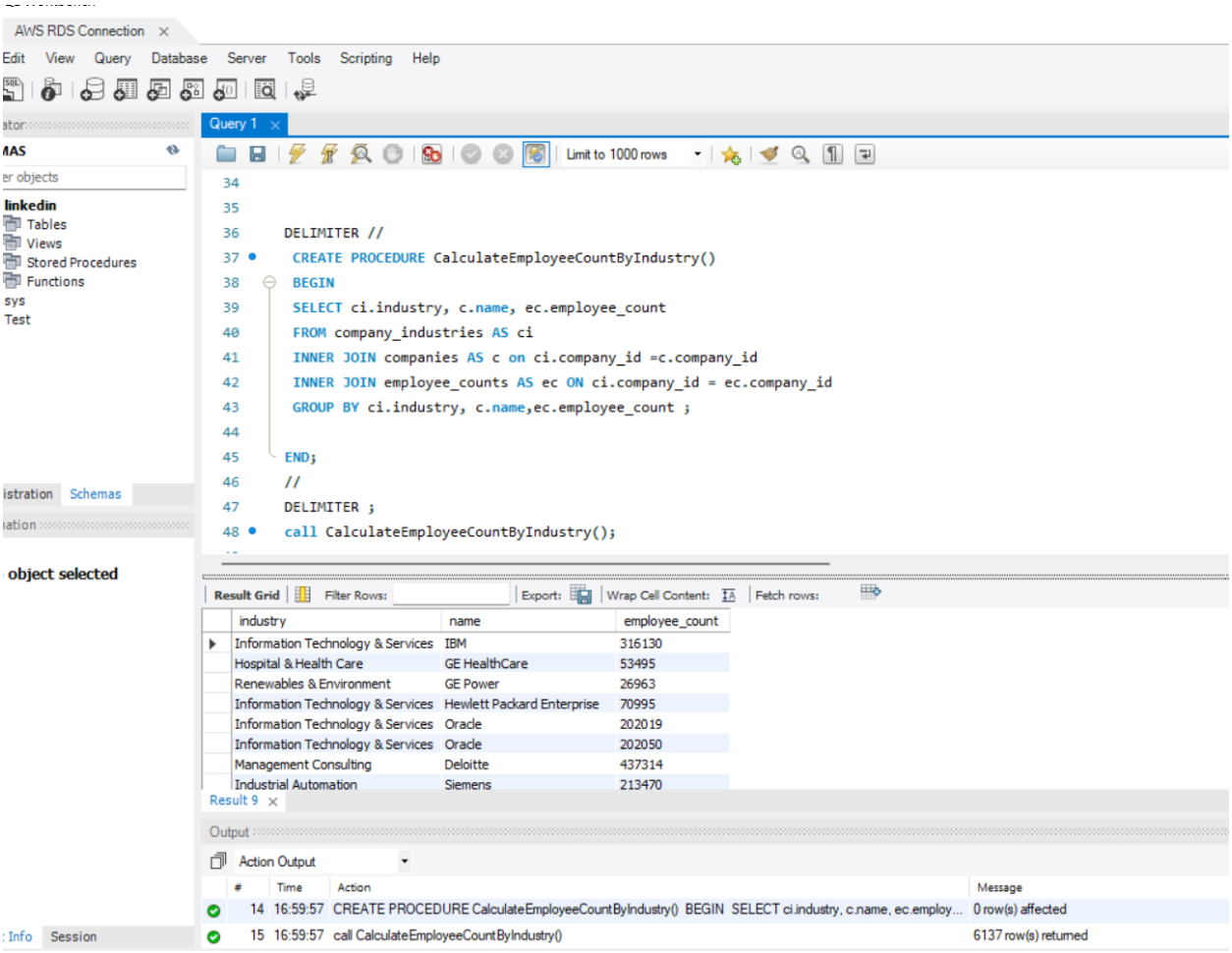
```
DELIMITER //
CREATE PROCEDURE GetCompanieswithspecificBenefit(BenefitType VARCHAR(100))
BEGIN
SELECT DISTINCT j.job_id, c.name AS company_name
FROM job_postings AS j
join companies as c on c.company_id = j.company_id
JOIN benefits AS b ON j.job_id = b.job_id
WHERE b.type = BenefitType;
END;
```

```
//
DELIMITER ;
call GetCompanieswithspecificBenefit('Vision insurance');
```



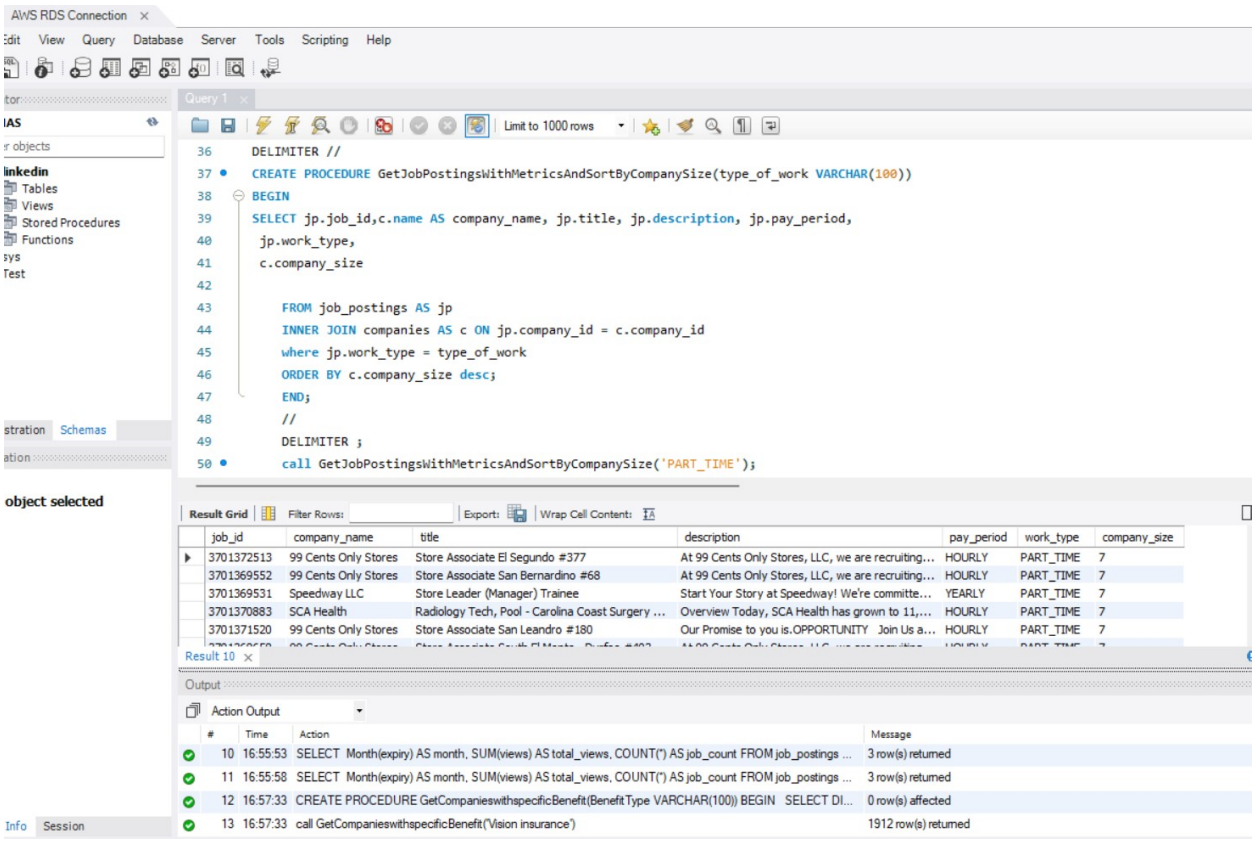
2. **Analysing Industry wise employee count**

```
DELIMITER //
CREATE PROCEDURE CalculateEmployeeCountByIndustry()
BEGIN
SELECT ci.industry, c.name, ec.employee_count
FROM company_industries AS ci
INNER JOIN companies AS c on ci.company_id =c.company_id
INNER JOIN employee_counts AS ec ON ci.company_id = ec.company_id
GROUP BY ci.industry, c.name,ec.employee_count ;
END;
//
DELIMITER ;
call CalculateEmployeeCountByIndustry();
```



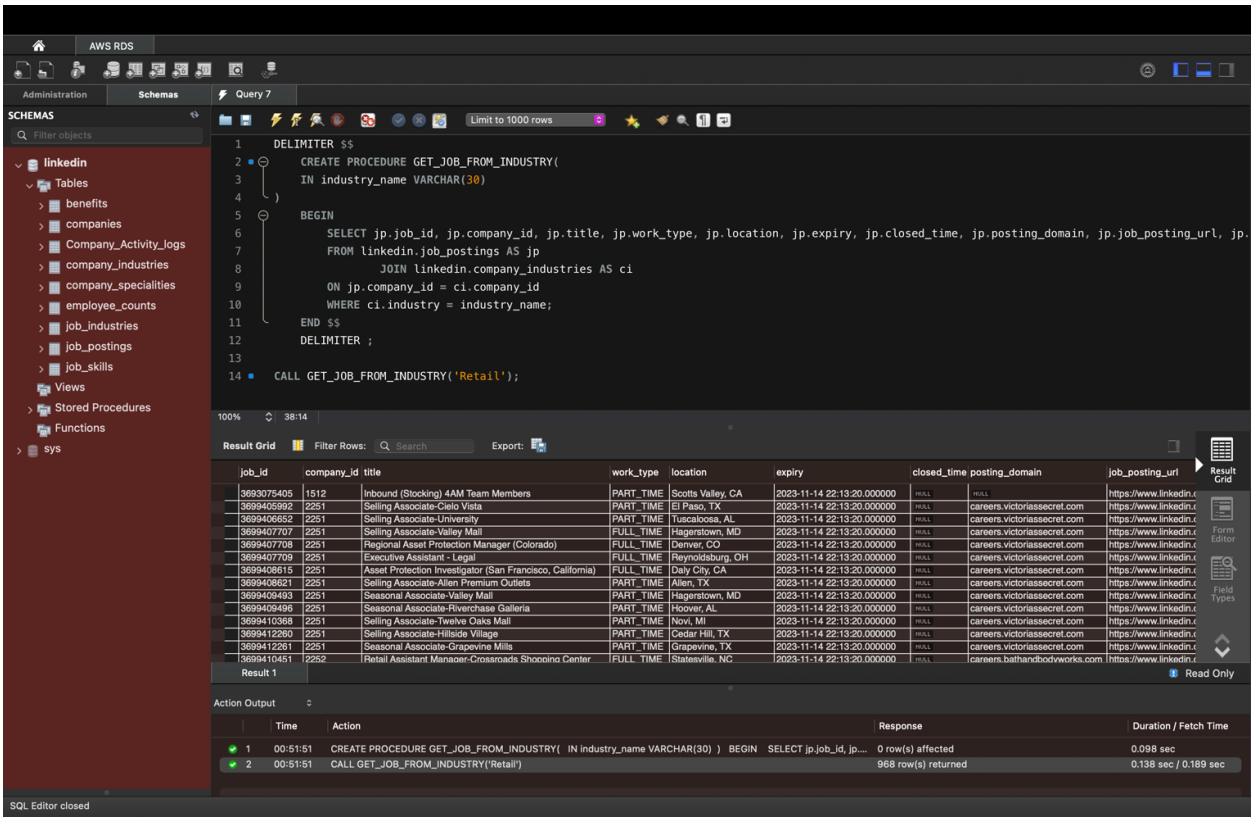
3. **Get Job Postings With Metrics by Company Size for given type of work**

```
DELIMITER //
CREATE PROCEDURE GetJobPostingsWithMetricsAndSortByCompanySize(type_of_work
VARCHAR(100))
BEGIN
SELECT jp.job_id,c.name AS company_name, jp.title, jp.description, jp.pay_period,
 jp.work_type,
 c.company_size

 FROM job_postings AS jp
 INNER JOIN companies AS c ON jp.company_id = c.company_id
 where jp.work_type = type_of_work
 ORDER BY c.company_size desc;
 END;
  //
 DELIMITER ;
call GetJobPostingsWithMetricsAndSortByCompanySize('PART_TIME');
```



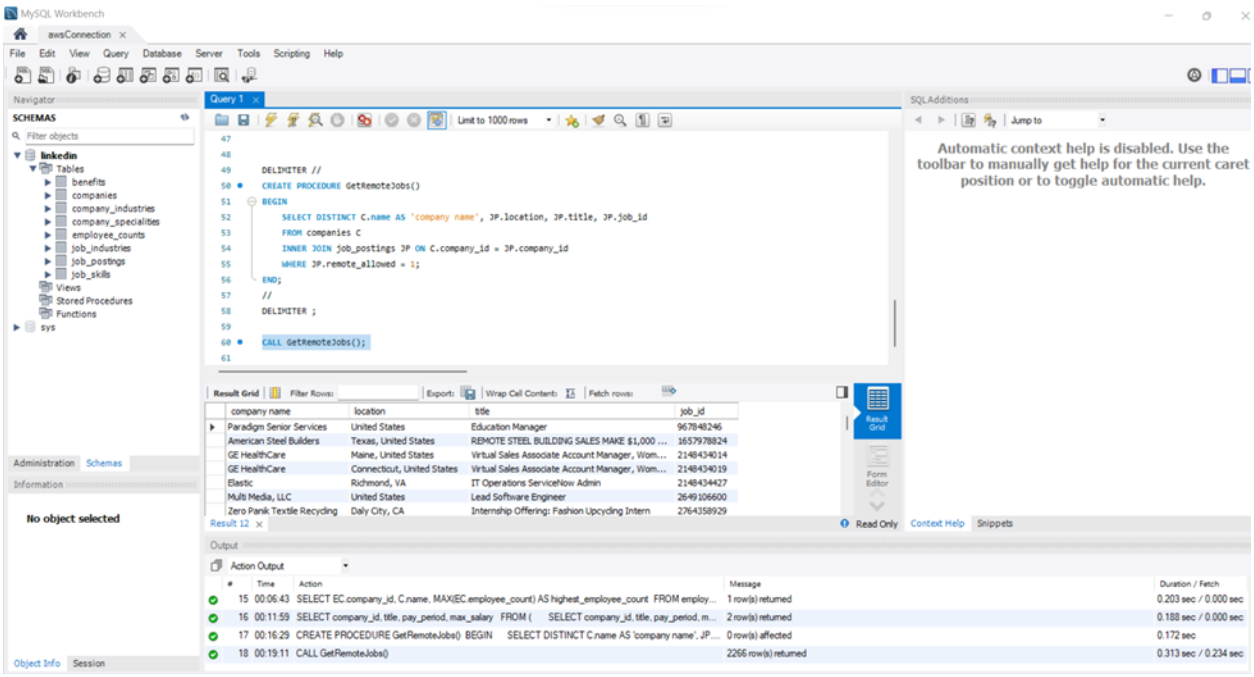
4. Find all jobs from a particular industry

```
DELIMITER $$
CREATE PROCEDURE GET_JOB_FROM_INDUSTRY(
IN industry_name VARCHAR(30))
BEGIN
SELECT jp.job_id, jp.company_id, jp.title, jp.work_type, jp.location, jp.expiry, jp.closed_time,
jp.posting_domain, jp.job_posting_url, jp.application_url
FROM linkedin.job_postings AS jp
JOIN linkedin.company_industries AS ci
ON jp.company_id = ci.company_id
WHERE ci.industry = industry_name;
END $$
DELIMITER ;

CALL GET_JOB_FROM_INDUSTRY('Retail');
```



5. **Display companies offering remote work.**

```
DELIMITER //
CREATE PROCEDURE GetRemoteJobs()
BEGIN
SELECT DISTINCT C.name AS 'company name', JP.location, JP.title, JP.job_id
FROM companies C
INNER JOIN job_postings JP ON C.company_id = JP.company_id
WHERE JP.remote_allowed = 1;
END;
//
DELIMITER ;
```

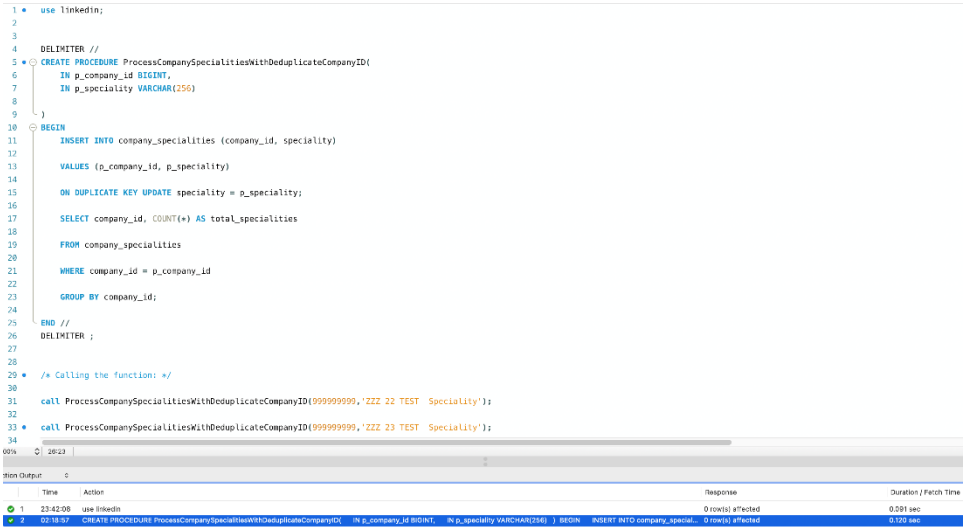


6. **Procedure to insert values to companies and company_specialities table and get the latest count of total specialities with its corresponding company_id.**

```
DELIMITER //

CREATE PROCEDURE ProcessCompanySpecialitiesWithDeduplicateCompanyID(
IN p_company_id BIGINT,
IN p_speciality VARCHAR(256))
BEGIN

INSERT INTO company_specialities (company_id, speciality)
VALUES (p_company_id, p_speciality)
ON DUPLICATE KEY UPDATE speciality = p_speciality;


SELECT company_id, COUNT(*) AS total_specialities
FROM company_specialities
WHERE company_id = p_company_id
GROUP BY company_id;
END //


DELIMITER ;
```

<u>Calling the function:</u>

call ProcessCompanySpecialitiesWithDeduplicateCompanyID(999999999,'ZZZ 22 TEST Speciality');



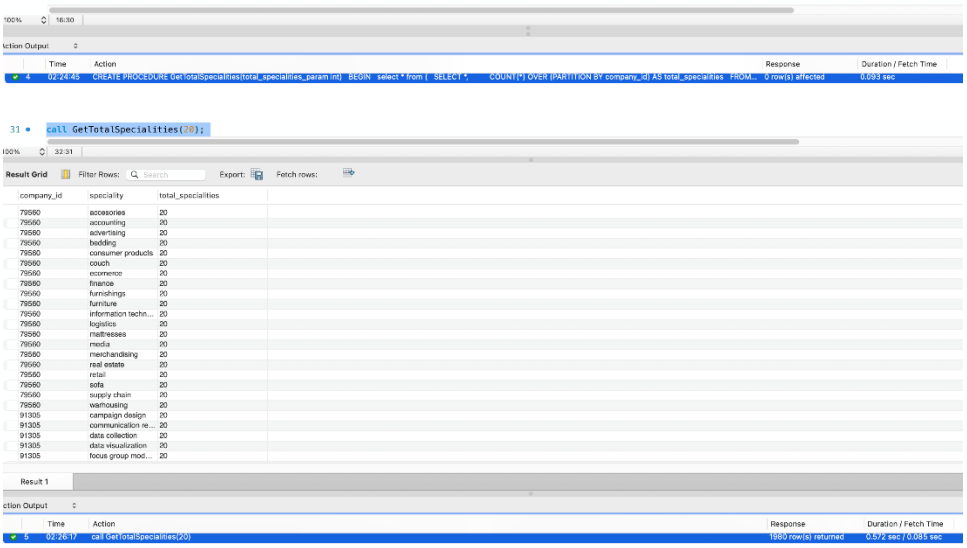
7. ***Procedure to get the exact user desired count as total_specialities_param to produce the company_id, speciality and total_specialities***

```
DELIMITER //
CREATE PROCEDURE GetTotalSpecialities(total_specialities_param int)
BEGIN
select * from (
SELECT *,
COUNT(*) OVER (PARTITION BY company_id) AS total_specialities
FROM company_specialities
order by (COUNT(*) OVER (PARTITION BY company_id)) desc
) A
where total_specialities =total_specialities_param;
    a.
END;
    b.
// DELIMITER ;
```

Calling the function:

call GetTotalSpecialities(20);

```sql
use linkedin;

DELIMITER //

CREATE PROCEDURE GetTotalSpecialities(total_specialities_param int)

BEGIN

select * from (

SELECT *,

      COUNT(*) OVER (PARTITION BY company_id) AS total_specialities

FROM company_specialities

order by (COUNT(*) OVER (PARTITION BY company_id)) desc

) A

where total_specialities =total_specialities_param;


END;


// DELIMITER ;
```



```sql
call GetTotalSpecialities(20);
```

company_id	speciality	total_specialities
79560	accesories	20
79560	accounting	20
79560	advertising	20
79560	bedding	20
79560	consumer products	20
79560	couch	20
79560	ecommerce	20
79560	finance	20
79560	furnishings	20
79560	furniture	20
79560	information techn...	20
79560	logistics	20
79560	mattresses	20
79560	media	20
79560	merchandising	20
79560	real estate	20
79560	retail	20
79560	sofa	20
79560	supply chain	20
79560	warhousing	20
91305	campaign design	20
91305	communication re...	20
91305	data collection	20
91305	data visualization	20
91305	focus group mod...	20

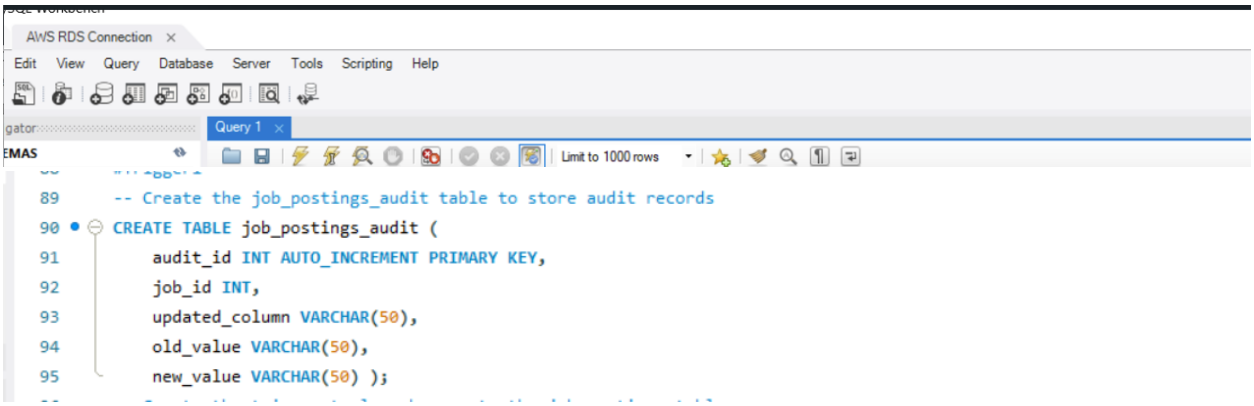
TRIGGERS:

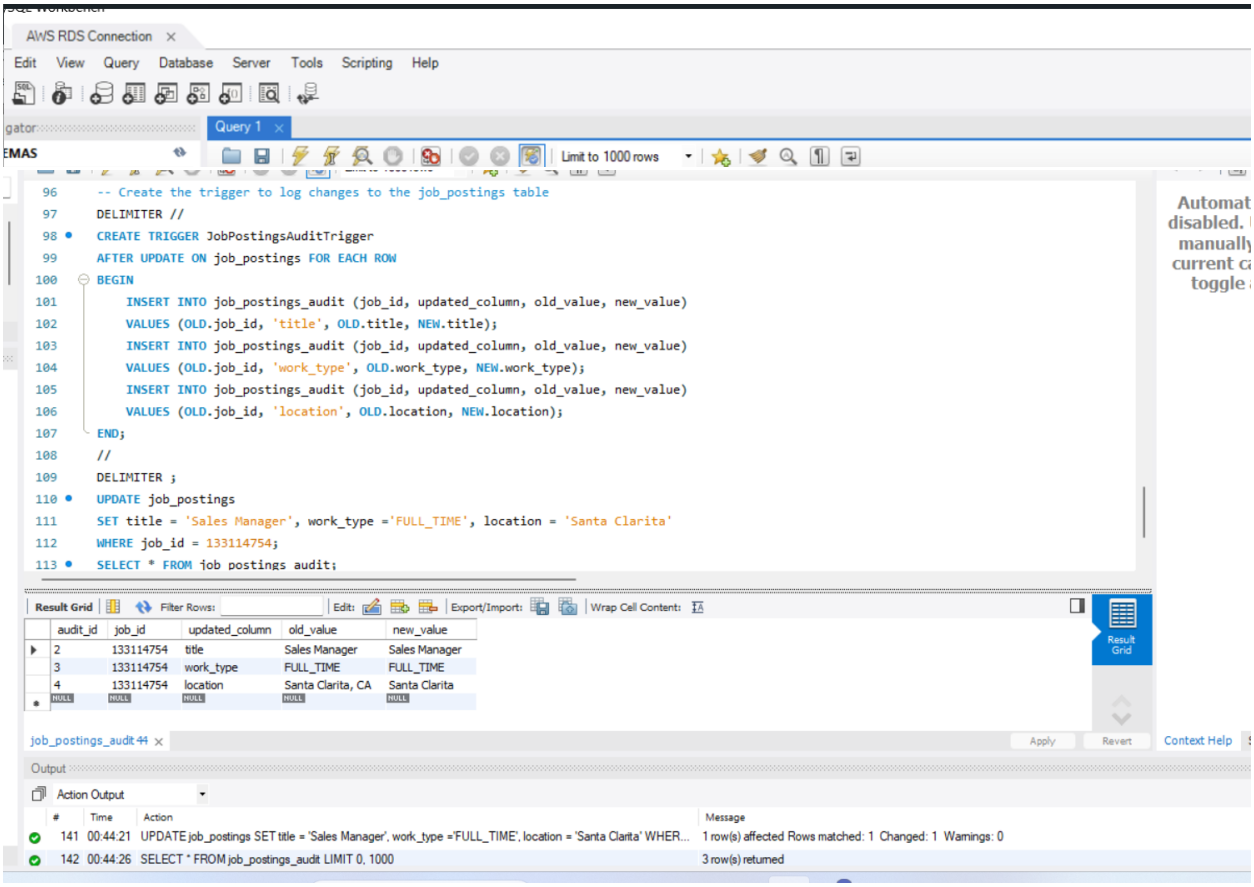
1. **Inserting value in Company activity log table after a deletion of company in companies table**

```
DELIMITER $$
CREATE TRIGGER Delete_job_posting_with_company
AFTER DELETE ON Companies FOR EACH ROW
BEGIN
UPDATE job_postings SET Company_id = NULL WHERE company_id = old.company_id;
INSERT INTO Company_Activity_logs VALUES ( old.Company_id, CONCAT('Row has been deleted from Job_postings for comapny ', old.Company_id));
END $$
DELIMITER ;
```

2 Creating a trigger to log changes to the job_postings table

```
DELIMITER //
CREATE TRIGGER JobPostingsAuditTrigger
AFTER UPDATE ON job_postings FOR EACH ROW
BEGIN
INSERT INTO job_postings_audit (job_id, updated_column, old_value, new_value)
VALUES (OLD.job_id, 'title', OLD.title, NEW.title);
INSERT INTO job_postings_audit (job_id, updated_column, old_value, new_value)
VALUES (OLD.job_id, 'work_type', OLD.work_type, NEW.work_type);
INSERT INTO job_postings_audit (job_id, updated_column, old_value, new_value)
VALUES (OLD.job_id, 'location', OLD.location, NEW.location);
END;
//
DELIMITER ;
UPDATE job_postings
SET title = 'Sales Manager', work_type ='FULL_TIME', location = 'Santa Clarita'
WHERE job_id = 133114754;
SELECT * FROM job_postings_audit;
```


AWS RDS Connection
Edit View Query Database Server Tools Scripting Help
Query 1
Limit to 1000 rows
-- Create the job_postings_audit table to store audit records
CREATE TABLE job_postings_audit (
audit_id INT AUTO_INCREMENT PRIMARY KEY,
job_id INT,
updated_column VARCHAR(50),
old_value VARCHAR(50),
new_value VARCHAR(50));


AWS RDS Connection
Edit View Query Database Server Tools Scripting Help
Query 1
Limit to 1000 rows
-- Create the trigger to log changes to the job_postings table
DELIMITER //
CREATE TRIGGER JobPostingsAuditTrigger
AFTER UPDATE ON job_postings FOR EACH ROW
BEGIN
INSERT INTO job_postings_audit (job_id, updated_column, old_value, new_value)
VALUES (OLD.job_id, 'title', OLD.title, NEW.title);
INSERT INTO job_postings_audit (job_id, updated_column, old_value, new_value)
VALUES (OLD.job_id, 'work_type', OLD.work_type, NEW.work_type);
INSERT INTO job_postings_audit (job_id, updated_column, old_value, new_value)
VALUES (OLD.job_id, 'location', OLD.location, NEW.location);
END;
//
DELIMITER ;
UPDATE job_postings
SET title = 'Sales Manager', work_type ='FULL_TIME', location = 'Santa Clarita'
WHERE job_id = 133114754;
SELECT * FROM job_postings_audit;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
audit_id job_id updated_column old_value new_value
2 133114754 title Sales Manager Sales Manager
3 133114754 work_type FULL_TIME FULL_TIME
4 133114754 location Santa Clarita, CA Santa Clarita
job_postings_audit 44
Apply
Revert
Context Help
Output
Action Output
Time Action Message
141 00:44:21 UPDATE job_postings SET title = 'Sales Manager', work_type ='FULL_TIME', location = 'Santa Clarita' WHER... 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
142 00:44:26 SELECT * FROM job_postings_audit LIMIT 0, 1000 3 row(s) returned