**SQL VIEWS (4):**

**SQL VIEWS:**

Two perspectives the views are going to be focused on:

* Employer Perspective
* Job Seeker Perspective.

VIEW 1: NAME: Sector Analysis

(Job Seeker Perspective)

SQL VIEW STRUCUTURE:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sector** | **Number of Jobs in sector** | **NumberOfCandidatesWithinSector** | **NumberOfCompaniesWitinSector** |

**CODE: USE OF: AGGREGATE FUNCTIONS, SQL JOINS, SUB QUERY**

3 TABLES : Job, JobSeeker\_Job, Qualification\_JobSeeker

CREATE VIEW SectorInfo AS

SELECT

Job.Sector

COUNT (Job.JobID) AS NumberOfJobsAvailable

FROM

JOB

GROUP BY

Job.Sector

CREATE VIEW SectorNumJobs AS

SELECT

Job.Sector AS Sector,

COUNT(DISTINCT Companies.CompanyID) AS NumberOfJobs

FROM

Job

INNER JOIN Companies ON Job.CompanyID = Companies.CompanyID

GROUP BY

Job.Sector

(Main View)

CREATE VIEW SectorAnalysis AS

SELECT

Job.Sector AS Sector,

(SELECT SectorInfo.NumberOfJobs FROM SectorInfo WHERE SectorInfo.Sector = Job.Sector) AS NumberOfJobs,

COUNT(JobSeeker\_Job.JobSeekerID) AS NumberOfCandidates,

(SELECT SectorNumJobs.NumberOfJobs FROM SectorNumJobs WHERE SectorNumJobs.Sector = Job.Sector) AS NumberOfCompaniesWithinSector

FROM

Job

INNER JOIN JobSeeker\_Job ON Job.JobID = JobSeeker\_Job.JobID

GROUP BY

Job.Sector

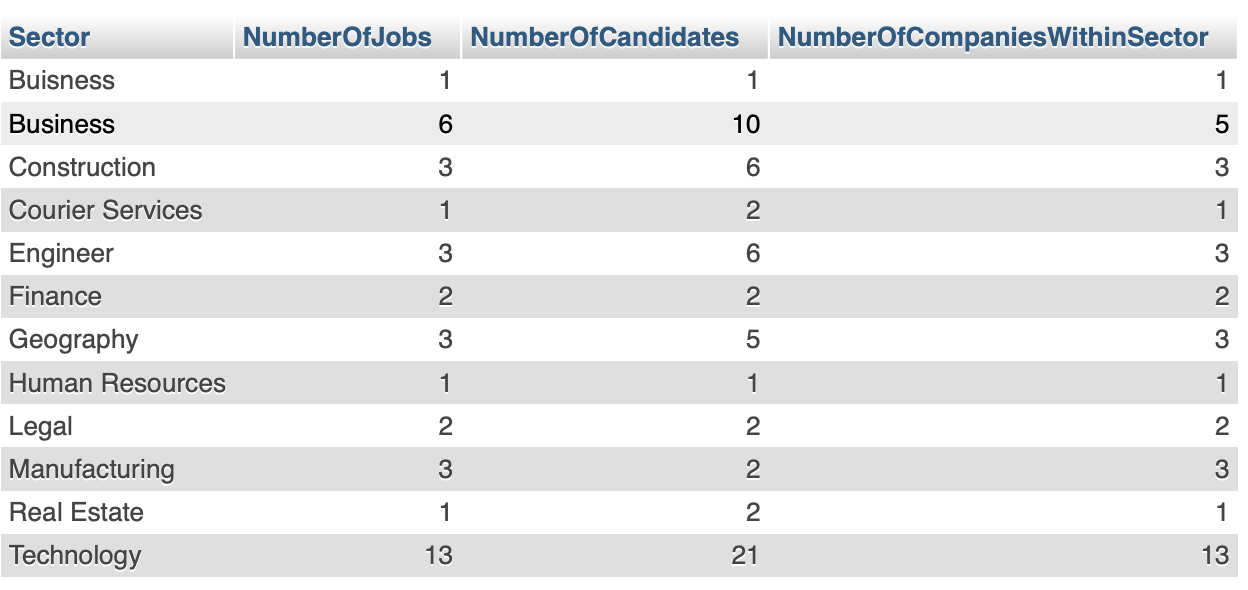
**View within view:**

This view uses a sub queries, within the sub queries other views are queried the views include (SectorInfo) and (SectorNumJobs). A view within a view was a structure I used in my query design and was required as SQL views did not support more than 1 result set, thus, did not support queries which had sub queries as this equates to > 1 result set.

**Purpose:**

This SQL view is useful for job seekers. This helps job seekers understand the industry, by analyzing each sector. The analysis is to derive the number of jobs in the sector, the number of candidates who have already matched a job within this sector, and the number of companies listing a job within this sector. This may help job seekers in understanding whether it is the right time to enter the job market (through filtering), and understand the competitiveness of the current market.

**Output:**



VIEW 2: NAME: Jobs closest to location

(Job Seeker Perspective)

SQL VIEW STRUCTURE:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Company Name** | **Job Name** | **Street** | **PostCode** | **City** | **Number of potential candidates** |

(Main View)

**CODE: AGGREGATE FUNCTIONS, SQL JOINS, WILD CARD**

3 TABLES: Company,Job, JobSeeker\_Job

CREATE VIEW JobClosestToLocation AS

SELECT

Companies.companyName,

Job.Name,

Job.Street,

Job.PostCode,

Job.City,

COUNT(JobSeeker\_Job.JobID) AS AlreadyPottentialCandidates

FROM Companies

INNER JOIN Job ON Companies.CompanyID = Job.CompanyID

LEFT JOIN JobSeeker\_Job ON Job.JobID = JobSeeker\_Job.JobID

WHERE

(Job.Street LIKE '%Malone%' AND Job.City = 'Belfast' AND Job.PostCode LIKE 'BT%')

GROUP BY

Job.Name

ORDER BY

JobSeeker\_Job.JobID DESC

**Purpose:**

This SQL view is to identify the jobs which are closest to the job seeker’s location. It could also be used to analyze the job by understanding its competitiveness through identifying the number of potential candidates. This SQL view uses the job seeker’s location and accordingly displays details including the company name, job name, street, Postcode, city, and number of potential candidates (Through filtering). The location of the job seeker can be changed by changing the arguments within the query. This can help job seekers with identifying appropriate jobs closest to the location.

**Output:**

A screenshot of a computer

Description automatically generated

VIEW 3: NAME: Jobs Open in March

(Job Seeker Perspective)

SQL VIEW STRUCTURE:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CompanyName** | **JobName** | **Sector** | **Salary** | **MinimumQualificationLevelAchieved** | **MaximumQualificationLevelAchieved** |

(Main View)

**CODE: FILTERING RECORDS, AGGREGATE FUNCTIONS, SQL JOINS**

3 TABLES: Companies, Job, JobSeekers\_Job

CREATE VIEW JobsOpenInMarch AS

SELECT

Companies.CompanyName,

Job.Name,

Job.Sector,

Job.Salary,

MIN(Qualification.QualificationLevel) AS MinimumQualificationLevelAchieved,

MAX(Qualification.QualificationLevel) AS MaximumQualificationLevelAchieved

FROM Companies

INNER JOIN Job ON Companies.CompanyID = Job.CompanyID

LEFT JOIN JobSeeker\_Job ON Job.JobID = JobSeeker\_Job.JobID

LEFT JOIN Qualification\_JobSeeker ON JobSeeker\_Job.JobSeekerID = Qualification\_JobSeeker.JobSeekerID

LEFT JOIN Qualification ON Qualification\_JobSeeker.QualificationID = Qualification.QualificationID

WHERE

(YEAR(Job.OpenDate) = 2024 AND MONTH(Job.OpenDate )= 3 AND DAY(Job.OpenDate) BETWEEN 0 AND 31)

GROUP BY

Companies.CompanyName

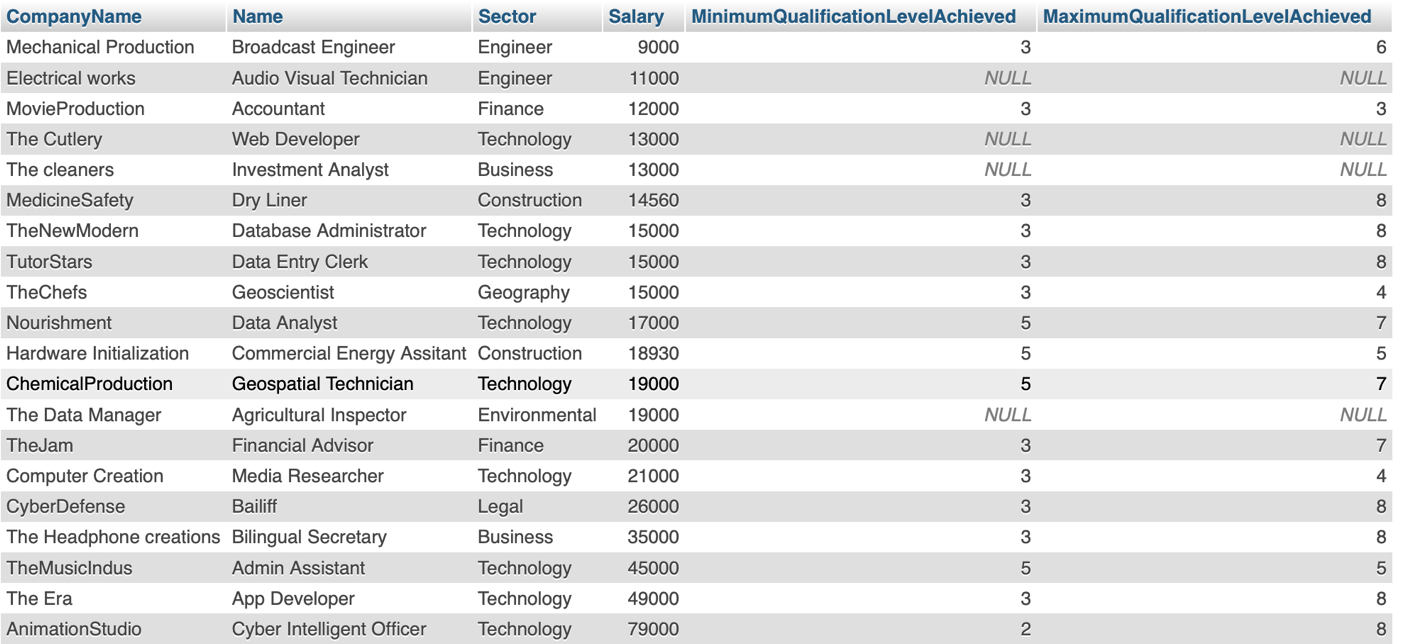
ORDER BY

Job.Salary ASC

**Purpose:**

This SQL view is identifying the jobs currently open in March (Through filtering). This query works by evaluating the job dates, where analyzed if within a certain range, if the condition is met, the company name, job name and maximum and minimum qualification level off the candidates matched with the job are shown. This would help users in identifying whether they have opportunities in joining a company, and whether their expertise is enough for a specific job role. Other dates can also be used, this is done through changing the queries arguments.

**Output:**



VIEW 4 NAME: Employee Analysis For Sector

(Employer Perspective)

SQL VIEW STRUCTURE:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Job Seeker First Name** | **Job Seeker Last Name** | **Age** | **NumberOfJobsMatched** | **Number of Qualifications** |

**CODE: AGGREGATE FUNCTIONS, SQL JOINS, FILTERING,**

4 TABLES: JobSeeker, JobSeeker\_Job, Job, Qualification\_JobSeeker

CREATE VIEW JobSeekerJobMatch AS

SELECT

JobSeeker\_Job.JobSeekerID,

COUNT(JobSeeker\_Job.JobID) AS NumberOfJobsPerJobSeeker

FROM

JobSeeker\_Job

GROUP BY

JobSeeker\_Job.JobSeekerID

CREATE VIEW QualificationsPerJobSeeker AS

SELECT

Qualification\_JobSeeker.JobSeekerID,

COUNT(Qualification\_JobSeeker.QualificationID) AS NumberOfQualificationsObtained

FROM

Qualification\_JobSeeker

GROUP BY

Qualification\_JobSeeker.JobSeekerID;

(Main View)

CREATE VIEW EmployeeAnalysisForSector AS

SELECT

JobSeeker.JobSeekerID AS JobSeekerIdentifier,

JobSeeker.FirstName,

JobSeeker.LastName,

ROUND((CURRENT\_DATE - DateOfBirth)/10000) AS Age,

(SELECT NumberOfJobsPerJobSeeker from JobSeekerJobMatch WHERE JobSeekerJobMatch.JobSeekerID =JobSeekerIdentifier) AS NumberOfJobsMatched,

(SELECT NumberOfQualificationsObtained FROM QualificationsPerJobSeeker WHERE QualificationsPerJobSeeker.JobSeeker = JobSeekerIdentifier) AS NumberOfQualifcationsObtained

FROM JobSeeker

INNER JOIN JobSeeker\_Job ON JobSeeker.JobSeekerID = JobSeeker\_Job.JobSeekerID

INNER JOIN Job ON JobSeeker\_Job.JobID = Job.JobID

INNER JOIN Qualification\_JobSeeker ON JobSeeker.JobSeekerID = Qualification\_JobSeeker.JobSeekerID

WHERE

Job.Sector = 'Technology'

GROUP BY

JobSeeker.JobSeekerID

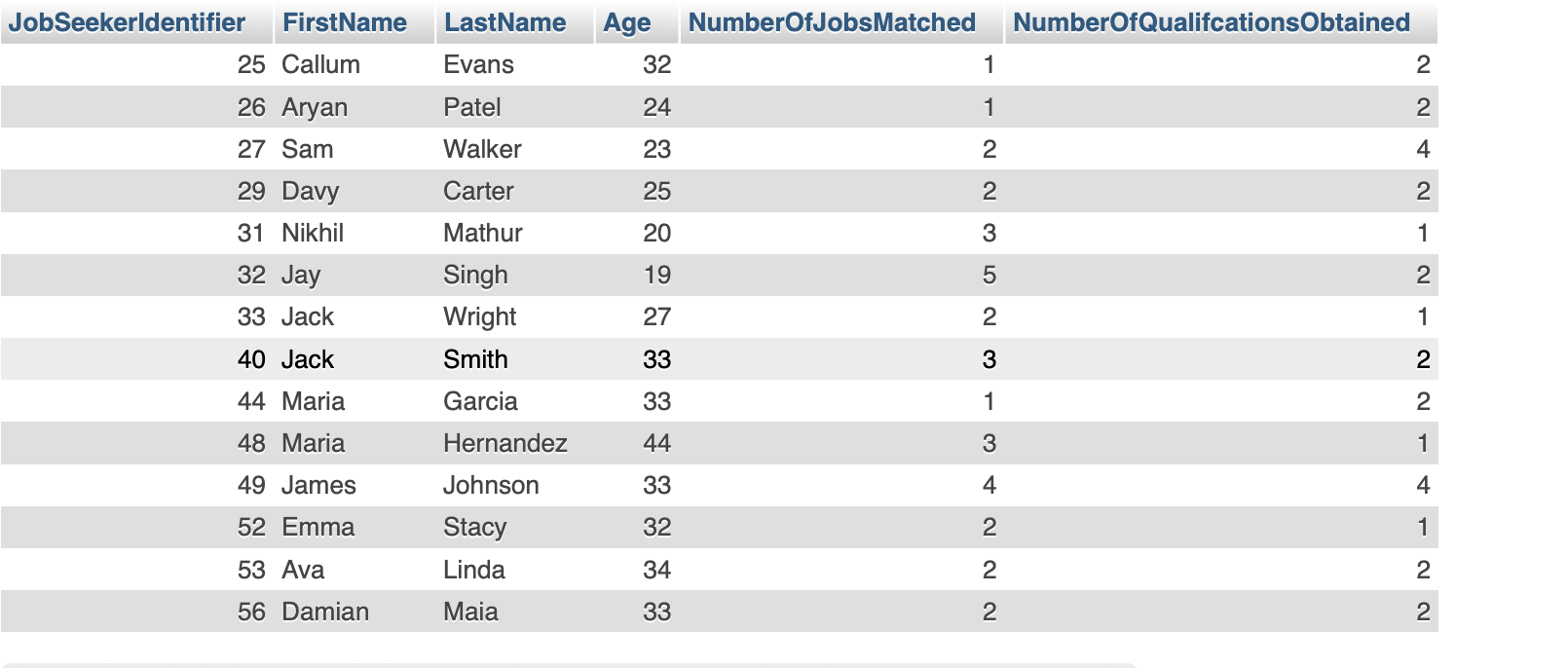
**View within view:**

Within this view, sub queries are used to query other views, the other views include (JobSeekerJobMatch) and (QualifcationPerJobSeeker). As SQL views do not support more than 1 result set, a query which contained a sub query would mean that there was > 1 result set. Hence, using this sub query alongside another view allowed for the view to be functional, with the relevant data displayed.

**Purpose:**

This SQL view will be useful from the employer ‘s perspective. Where the current job seekers in the market are analyzed, where the number of jobs matches of a candidate are displayed, alongside their age and number of qualifications a candidate has obtained. To make this more personalized for an employer, the job sector is used for filtering purposes where the employer enters the specific sector, and the relevant job seekers are displayed. The employers from here can see which potential candidates are best suited and are off benefit for a particular business.

**Output:**



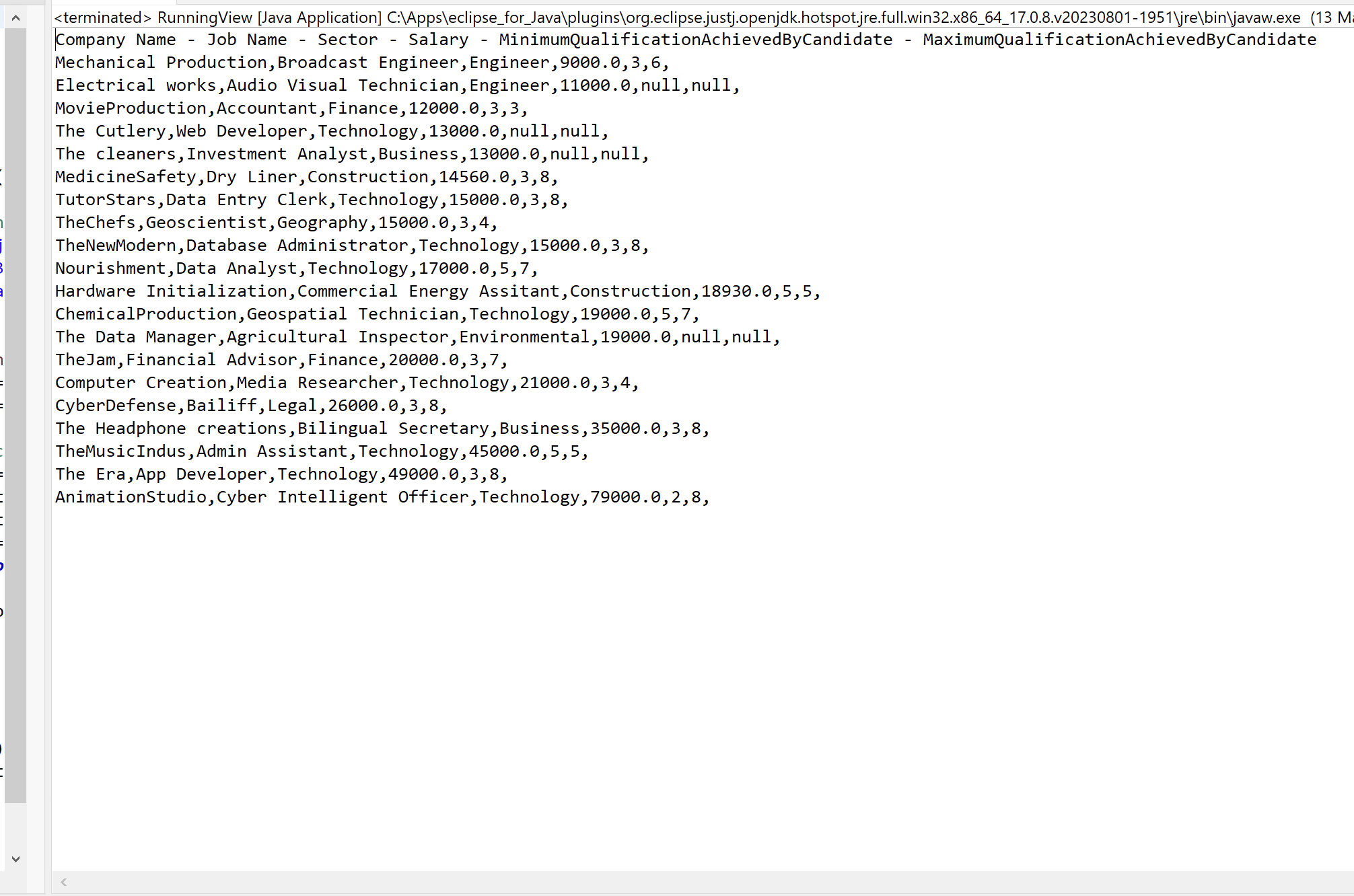
**Java connection:**

**Displaying one view (JobsOpenInMarch View):**

**Code:**

A screenshot of a computer code

Description automatically generated

**Output:**

**Inserting data into Qualification table:**

**Code:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated**Inserting data into all attributes (columns):**

**Output(Data inserted successfully):**

**A screenshot of a computer

Description automatically generated**

**Inserting into all not null columns (not inserting into attributes declared null in “Required”):**

**A screenshot of a computer

Description automatically generated**

**Output (Data inserted successfully):**

A screenshot of a computer

Description automatically generated