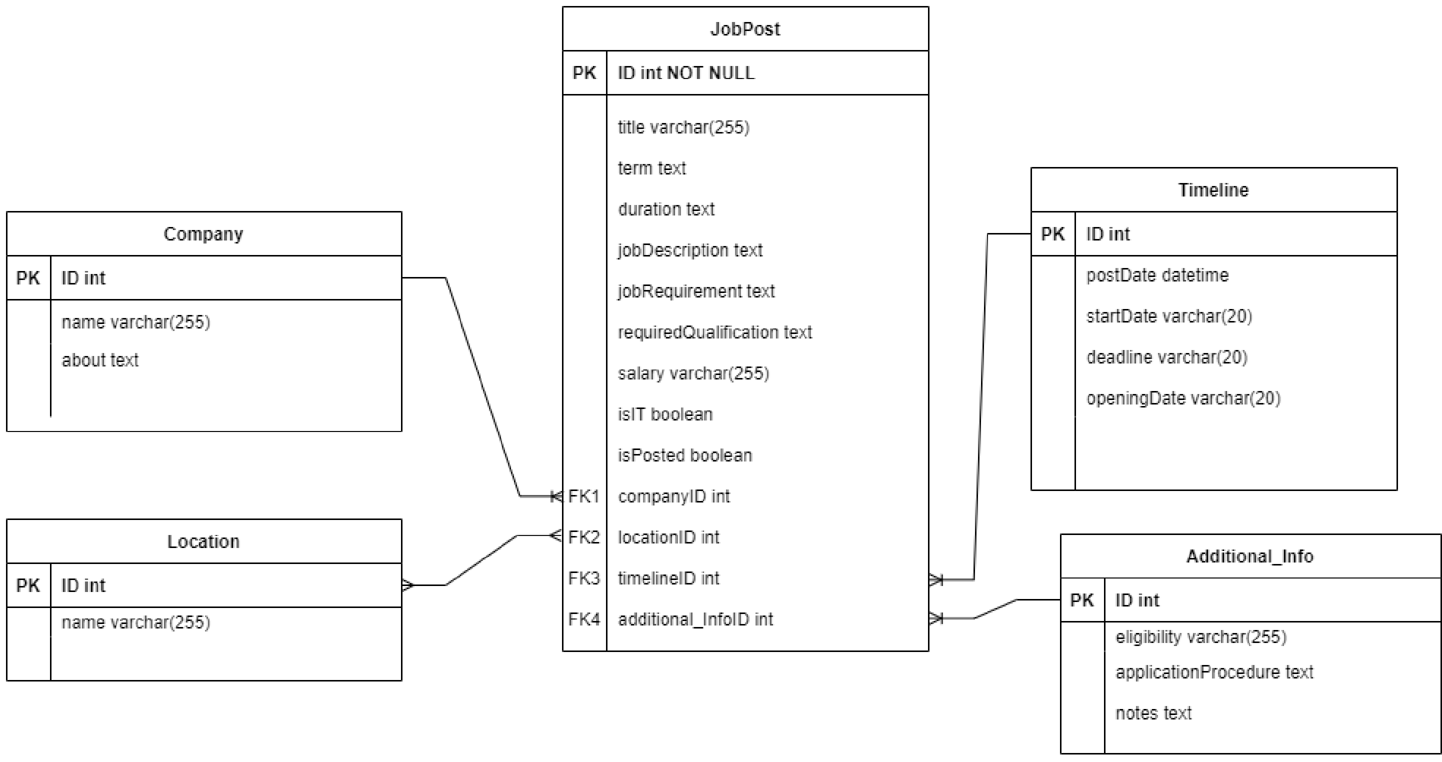
Database Design - Final Project Part 2

This database design document provides insight into the systematic structuring of our database tables, incorporating thorough normalization processes to establish the foundation for the Job Connect - Recruitment Portal. Our project is dedicated to transforming the landscape of online job discovery and recruitment by creating an innovative platform. The primary focus is on enhancing user engagement through personalized and informed job recommendations, making customization a central tenet of our approach.The design of our database tables follows normalization procedures, ensuring data integrity, minimizing redundancy, and optimizing for efficient data retrieval. Each table within the database is crafted to represent specific entities and their relationships accurately. The utilization of primary keys, foreign keys, and constraints guarantees the consistency and reliability of our data.

Create Conceptual Diagram/Schema for Online Job Postings Database



In the Job Connect - Recruitment Portal database project, the conceptual schema comprises five interconnected tables: JobPost, Company, Location, Timeline, and Additional_Info.

Entities:

- **JobPost**: Represents individual job postings, each linked to a specific company, location, timeline, and additional information.
- **Company**: Represents companies posting job opportunities.
- **Location**: Represents job locations.
- **Timeline**: Represents timelines associated with job postings.
- **Additional_Info**: Contains additional information related to job postings.

The use of primary keys, foreign keys, and constraints in database tables is fundamental to maintaining data integrity, ensuring accurate relationships between tables, and enforcing business rules. Each table consists of primary key, relevant constraints and foreign keys. Primary keys in the entities are used for uniquely identifying records, indexing and maintaining data integrity. For example in our schema and conceptual diagram, the ‘ID’ column serves as the primary key in each table. While foreign keys in the entities are used to establish relationships and ensure referential integrity. It ensures to maintain data consistency in the database tables. For example, in our schema the ‘JobPost’ table has 4 foreign key columns which are ‘companyID’, ‘locationID’, ‘timelineID’, and ‘additional_infoID’. They reference the respective primary keys in the Company, Location, Timeline, and Additional_Info tables, establishing relationships between them. The constraints in the tables help to enforce business rules and ensure data quality. There are different types of constraints which are NOT NULL, UNIQUE, CHECK, DEFAULT, PRIMARY KEY and FOREIGN KEY. For example, in our schema the company table, the ‘name’ column has a UNIQUE NOT NULL constraint. This ensures that each company has a unique and non-null name.

Relation Types:

1. **Company - JobPost (One-to-Many):**
   This relationship signifies that one company can be associated with multiple job posts within the JobPost table. This is represented by a foreign key (companyID) in the JobPost table that references the primary key (ID) in the Company table.

2. **Location - JobPost (Many-to-Many):**
   This relationship reflects the versatility in job locations. A location can be associated with multiple job posts, and conversely, a job post can be available in multiple locations. This is represented through a direct connection between Location(ID) and JobPost(locationID).

3. **Timeline - JobPost (One-to-Many):**
   This relationship indicates that one timeline can be associated with multiple job posts. This is represented by a foreign key (timelineID) in the JobPost table, referencing the primary key (ID) in the Timeline table.

4. **Additional_Info - JobPost (One-to-Many):**
   This relationship signifies that one set of additional information can be associated with multiple job posts. This is indicated by a foreign key (additional_infoID) in the JobPost table, referencing the primary key (ID) in the Additional_Info table.

JobPost:

The "JobPost" is a transactional table containing distinct entries for each job posting made by all the companies, with the "Id" serving as the primary key. It also contains foreign keys referencing all the other tables which can be used to fetch additional data like location, and application deadline for all the job posts.

Columns:-

Id: The "Id" is set as an auto-incremented integer data type.
Title: The "Title" column, defined as varchar, is designated to store the name of the job position listed in the job post. It does not allow null values.
Term: The “Term” column, defined as text, is designated to store the nature of the job like full-time, part-time, contract basis, etc.
Duration: The “Duration” column, defined as text, is designated to store the duration of the employment.
Jobdescription: The “jobdescription” column, defined as text, is designated to store Details about the job.
Jobrequirement: The “jobrequirement ” column, defined as text, is designated to store duties that need be be fulfilled by the candidate.
Requiredqualification: The “requiredqualification ” column, defined as text, is designated to store the minimum qualifications required for the job.
Salary: The “salary” column, defined as varchar, is designated to store the salary

offered for a particular position.
Isit: The “isit ” column, defined as a boolean, indicates whether is job post is related to IT field or not.
Isposted: The “isposted” column, defined as a boolean, indicates whether is job post is Approved by HR manager and posted on job portal or not.
Companyid: The “companyid ” column, defined as an integer, is a foreign key referencing the “id” column of the “company” table.
Locationid: The “locationid ” column, defined as an integer, is a foreign key referencing the “id” column of the “location” table.
Timelineid: The “timelineid” column, defined as an integer, is a foreign key referencing the “id” column of the “timeline” table.
Additional_infoid*:* The “additional_infoid” column, defined as an integer, is a foreign key referencing the “id” column of the “Additional_info” table

Company:

The "Company" table contains distinct entries for each company, with the "Id" serving as the primary key for mapping to other tables.

Columns*:-*

***Id*:** The "Id" is set as an auto-incremented integer data type.
***Name*:** The "Name" column, defined as varchar, is designated to store the unique and non-null names of companies
***About*:** the table features an "About" column with a text data type, intended for recording detailed information or metadata about each company.

Location:

The "Location" table contains distinct entries for each location where job is available, with the "Id" serving as the primary key for mapping to other tables.

Columns*:-*

***Id*:** The "Id" is set as an auto-incremented integer data type with primary key.
***Name*:** The "Name" column, defined as varchar, is designated to store the unique and non-null location names for the job post.

Timeline:

The "Timeline" table contains distinct entries for each timeline associated with job postings, with the "ID" serving as the primary key for mapping to other tables.

Columns*:-*

ID: The "ID" is set as an auto-incremented integer data type with the primary key constraint. It serves as a unique identifier for each timeline record.
postDate: It stores the date when the job post is made. Data Type: varchar(50)
startDate: It represents the start date of the job Data Type: text
Deadline: It indicates the deadline for submitting job applications. Data Type: text
openingDate: It stores the opening date of the job announcement i.e the date when the job position opens for the applications. Data Type: text

Additional_info:

The "Additional_Info" table contains distinct entries for additional information related to job postings, with the "ID" serving as the primary key for mapping to other tables.

Columns*:-*

ID: The "ID" is set as an auto-incremented integer data type with the primary key constraint. It serves as a unique identifier for each additional information record.
eligibility: It stores information about the eligibility criteria for the job. Data Type: text
applicationProcedure: This column captures details regarding the application procedures for the job. Data Type: text
Notes: This attribute provides a space for any additional notes or information relevant to the job posting. Data Type: text

Database Constraints:

Database constraints are regulations applied to table columns to uphold data integrity, accuracy, and reliability. They prevent inconsistencies, ensuring data meets defined conditions. Constraints enforce business rules, maintain table relationships, and uphold data quality standards.
Some common types of constraints are Primary Key Constraints, Foreign Key Constraints, Unique Constraints, NOT NULL Constraints, Default Constraints and Check Constraints.

Primary Key Constraint:

It ensures that each record in a table has a unique identifier, preventing duplicate or null values in the specified column or combination of columns.
In our Job Connect - Recruitment Portal database schema, primary key constraint is applied to the Company, Location, Timeline, Additional_Info, and JobPost tables.

‘ID’ column is the primary key constraint in the Company, Location, Timeline, Additional_Info, and JobPost tables. This column has a property of auto-increment whenever a new row is added to a table.

Foreign Key Constraint:
It establishes relationships between tables by linking the primary key of one table to the foreign key of another, enforcing referential integrity.
In our Job Connect - Recruitment Portal database schema, the ‘companyID’, ‘locationID’, ‘timelineID’ , and ‘additional_infoID’ columns in the JobPost table is the foreign key constraint which references the corresponding primary keys in Company, Location, Timeline and Additional_Info tables.

Unique Constraint:
It ensures that all values in a specified column or combination of columns are unique, preventing the insertion of duplicate values.
In our Job Connect - Recruitment Portal database schema, the ‘Name’ columns in the Company table and Location table have unique constraints to ensure that each company and each location has a unique name.

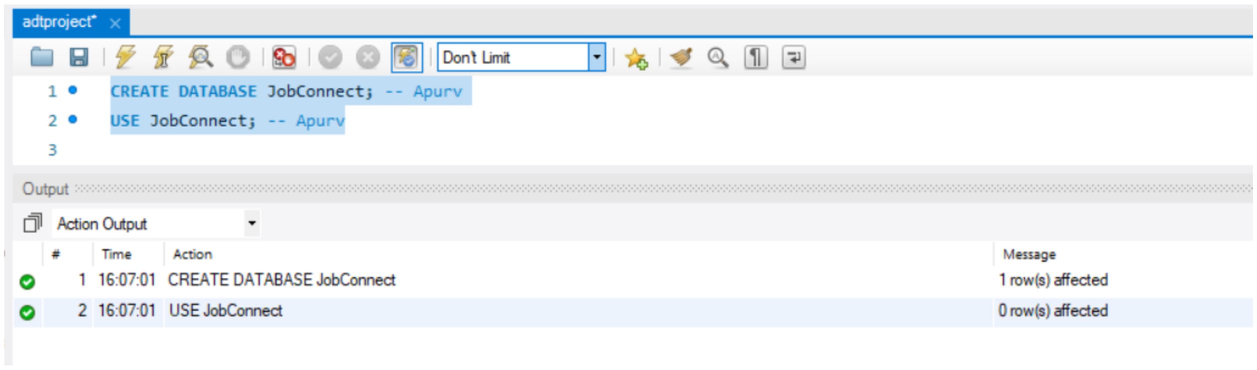
NOT NULL Constraint:
It ensures that a specified column does not contain null (empty) values, maintaining data integrity.
In our Job Connect - Recruitment Portal, NOT NULL constraint is applied to columns like Name in the Company table, Name in the Location table, postDate in the Timeline table, and title in the JobPost table.

Default Constraint:
It provides a default value for a column when no value is specified during an INSERT operation.
In Job Connect - Recruitment Portal, this constraint is applied to the ‘isPosted’ column where a default value is set to 0 in case no value is specified in insert queries.

Write code to create a database and build queries. Your task is to create a reproducible code.

create db:

```
CREATE DATABASE JobConnect; -- Apurv
USE JobConnect; -- Apurv
```



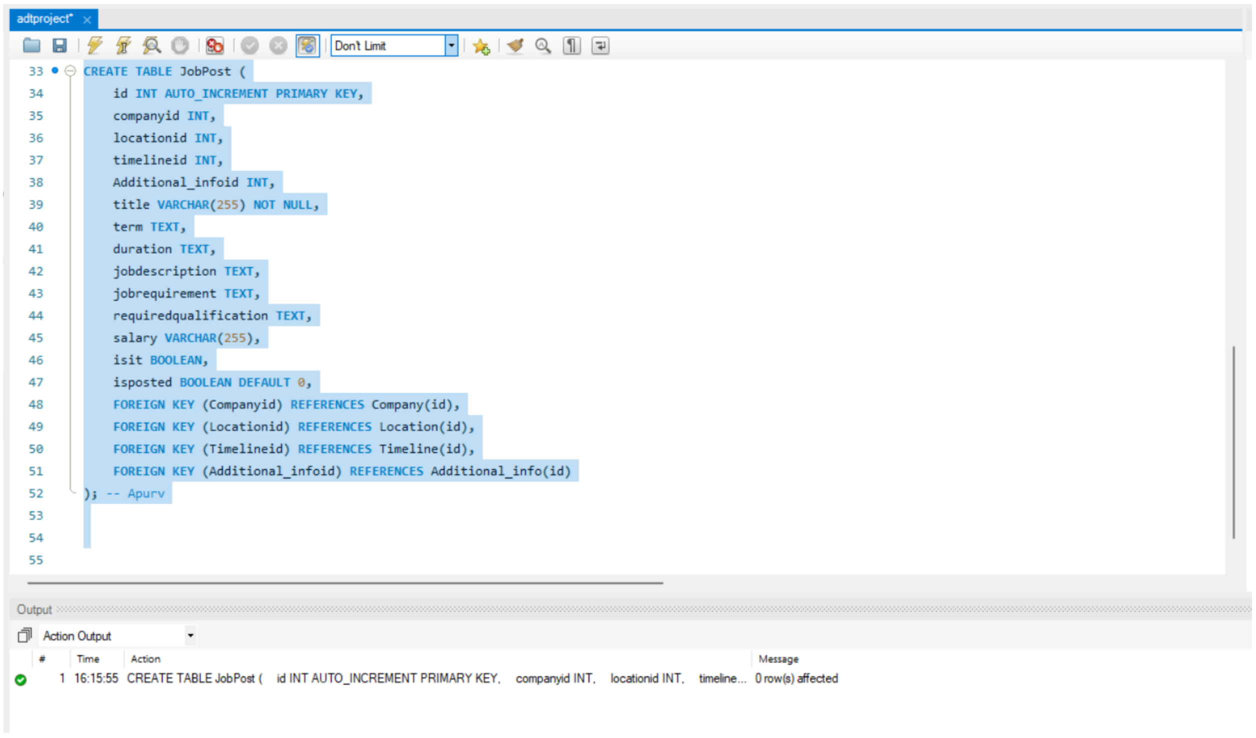
```
CREATE DATABASE JobConnect; -- Apurv
USE JobConnect; -- Apurv
```

```
#  Time      Action                              Message
1  16:07:01  CREATE DATABASE JobConnect          1 row(s) affected
2  16:07:01  USE JobConnect                      0 row(s) affected
```

create tables, data types and keys:

```
-- Apurv
CREATE TABLE JobPost (
    id INT AUTO_INCREMENT PRIMARY KEY,
    companyid INT,
    locationid INT,
    timelineid INT,
    Additional_infoid INT,
    title VARCHAR(255) NOT NULL,
    term TEXT,
    duration TEXT,
    jobdescription TEXT,
    jobrequirement TEXT,
    requiredqualification TEXT,
    salary VARCHAR(255),
    isIT BOOLEAN,
    isPosted BOOLEAN DEFAULT 0,
    FOREIGN KEY (Companyid) REFERENCES Company(id),
    FOREIGN KEY (Locationid) REFERENCES Location(id),
    FOREIGN KEY (Timelineid) REFERENCES Timeline(id),
    FOREIGN KEY (Additional_infoid) REFERENCES Additional_info(id)
```

);

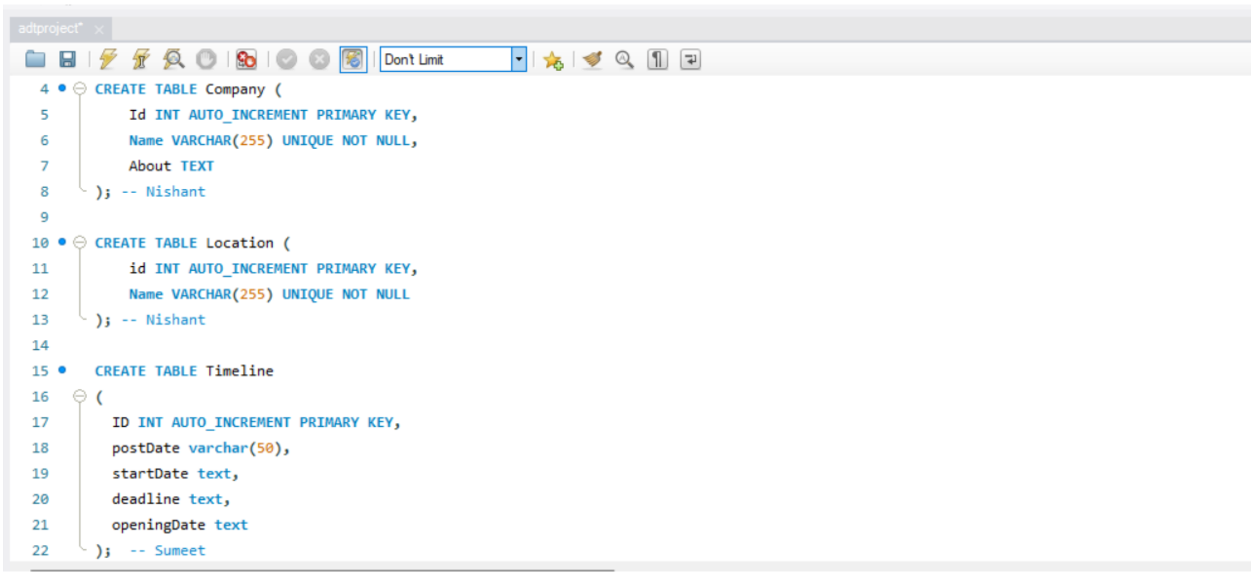


```
CREATE TABLE JobPost (
    id INT AUTO_INCREMENT PRIMARY KEY,
    companyid INT,
    locationid INT,
    timelineid INT,
    Additional_infoid INT,
    title VARCHAR(255) NOT NULL,
    term TEXT,
    duration TEXT,
    jobdescription TEXT,
    jobrequirement TEXT,
    requiredqualification TEXT,
    salary VARCHAR(255),
    isit BOOLEAN,
    isposted BOOLEAN DEFAULT 0,
    FOREIGN KEY (Companyid) REFERENCES Company(id),
    FOREIGN KEY (Locationid) REFERENCES Location(id),
    FOREIGN KEY (Timelineid) REFERENCES Timeline(id),
    FOREIGN KEY (Additional_infoid) REFERENCES Additional_info(id)
); -- Apurv
```

```
Output
Action Output
   #  Time      Action                                                                                 Message
✓  1  16:15:55  CREATE TABLE JobPost (    id INT AUTO_INCREMENT PRIMARY KEY,    companyid INT,    locationid INT,    timeline...  0 row(s) affected
```

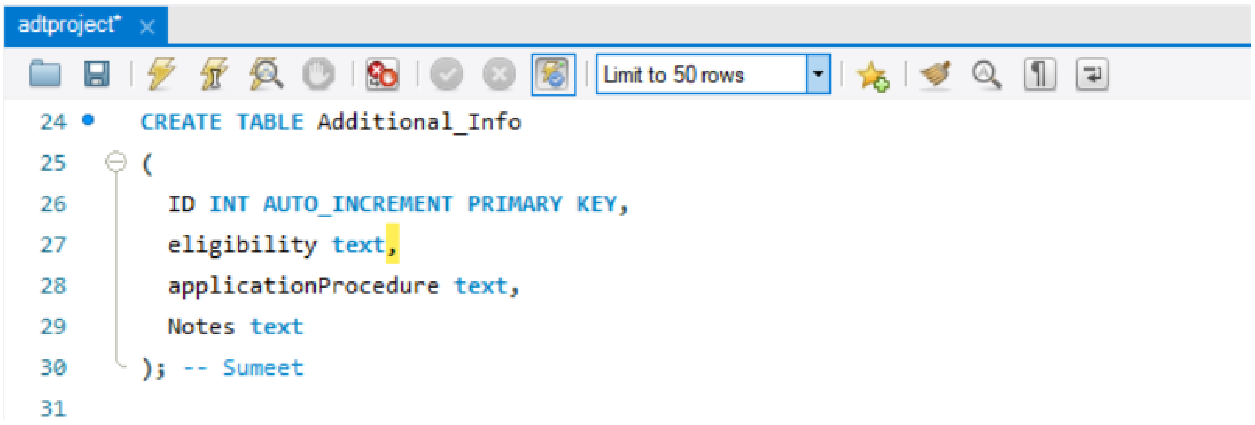
```
-- Nishant
CREATE TABLE Company (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) UNIQUE NOT NULL,
    About TEXT
);


-- Nishant
CREATE TABLE Location (
    id INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) UNIQUE NOT NULL
);

-- Sumeet
CREATE TABLE Timeline
(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  postDate varchar(50),
  startDate text,
  deadline text,
  openingDate text
```

);



```
CREATE TABLE Company (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) UNIQUE NOT NULL,
    About TEXT
); -- Nishant

CREATE TABLE Location (
    id INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) UNIQUE NOT NULL
); -- Nishant

CREATE TABLE Timeline
(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  postDate varchar(50),
  startDate text,
  deadline text,
  openingDate text
);  -- Sumeet
```

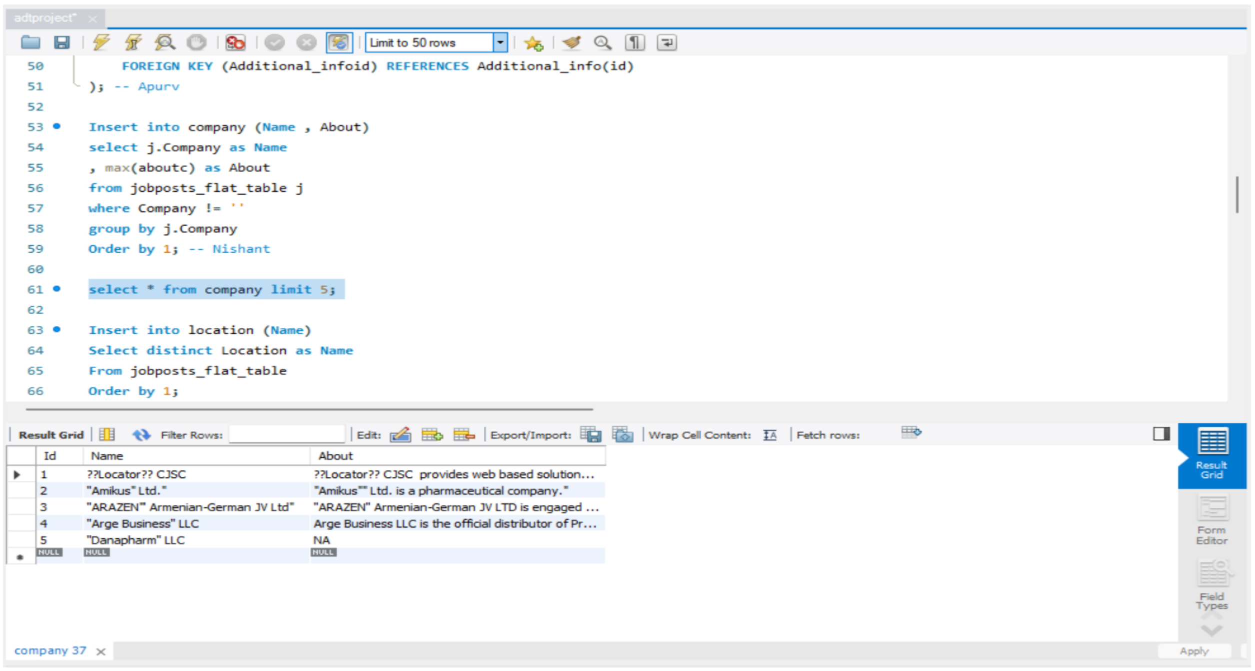
```
-- Sumeet
CREATE TABLE Additional_Info
(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  eligibility text,
  applicationProcedure text,
  Notes text
);
```



```
CREATE TABLE Additional_Info
(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  eligibility text,
  applicationProcedure text,
  Notes text
); -- Sumeet
```

Insert data:

```
– For company table -- Nishant
Insert into company (Name , About)
select j.Company as Name
, max(aboutc) as About
from jobposts_flat_table j
where Company != ''
group by j.Company
Order by 1;
```

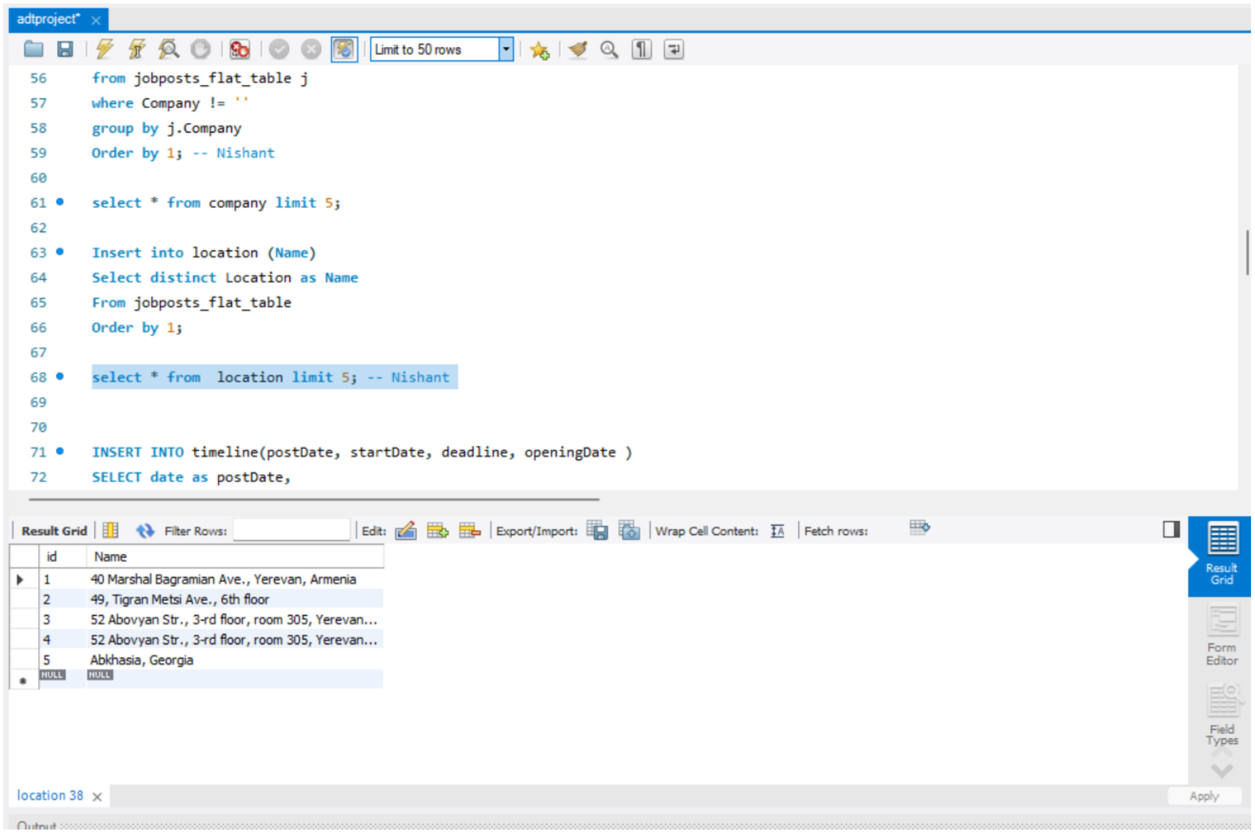


```
    FOREIGN KEY (Additional_infoid) REFERENCES Additional_info(id)
); -- Apurv

Insert into company (Name , About)
select j.Company as Name
, max(aboutc) as About
from jobposts_flat_table j
where Company != ''
group by j.Company
Order by 1; -- Nishant

select * from company limit 5;

Insert into location (Name)
Select distinct Location as Name
From jobposts_flat_table
Order by 1;
```

Id	Name	About
1	??Locator?? CJSC	??Locator?? CJSC provides web based solution...
2	"Amikus" Ltd."	"Amikus"" Ltd. is a pharmaceutical company."
3	"ARAZEN"" Armenian-German JV Ltd"	"ARAZEN" Armenian-German JV LTD is engaged ...
4	"Arge Business" LLC	Arge Business LLC is the official distributor of Pr...
5	"Danapharm" LLC	NA
NULL	NULL	NULL

company 37

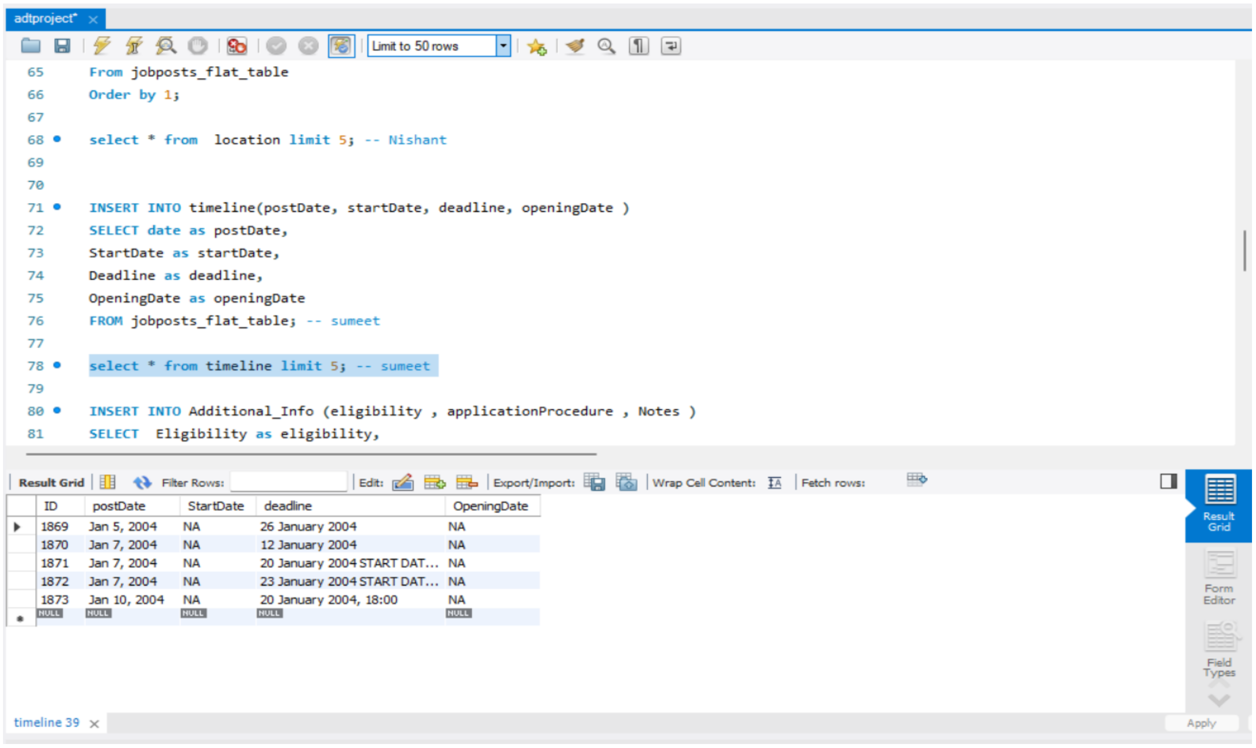
```
– For location table -- Nishant
Insert into location (Name)
Select distinct Location as Name
From jobposts_flat_table
Order by 1;
```



```
from jobposts_flat_table j
where Company != ''
group by j.Company
Order by 1; -- Nishant

select * from company limit 5;

Insert into location (Name)
Select distinct Location as Name
From jobposts_flat_table
Order by 1;

select * from  location limit 5; -- Nishant


INSERT INTO timeline(postDate, startDate, deadline, openingDate )
SELECT date as postDate,
```

id	Name
1	40 Marshal Bagramian Ave., Yerevan, Armenia
2	49, Tigran Metsi Ave., 6th floor
3	52 Abovyan Str., 3-rd floor, room 305, Yerevan...
4	52 Abovyan Str., 3-rd floor, room 305, Yerevan...
5	Abkhasia, Georgia
NULL	NULL

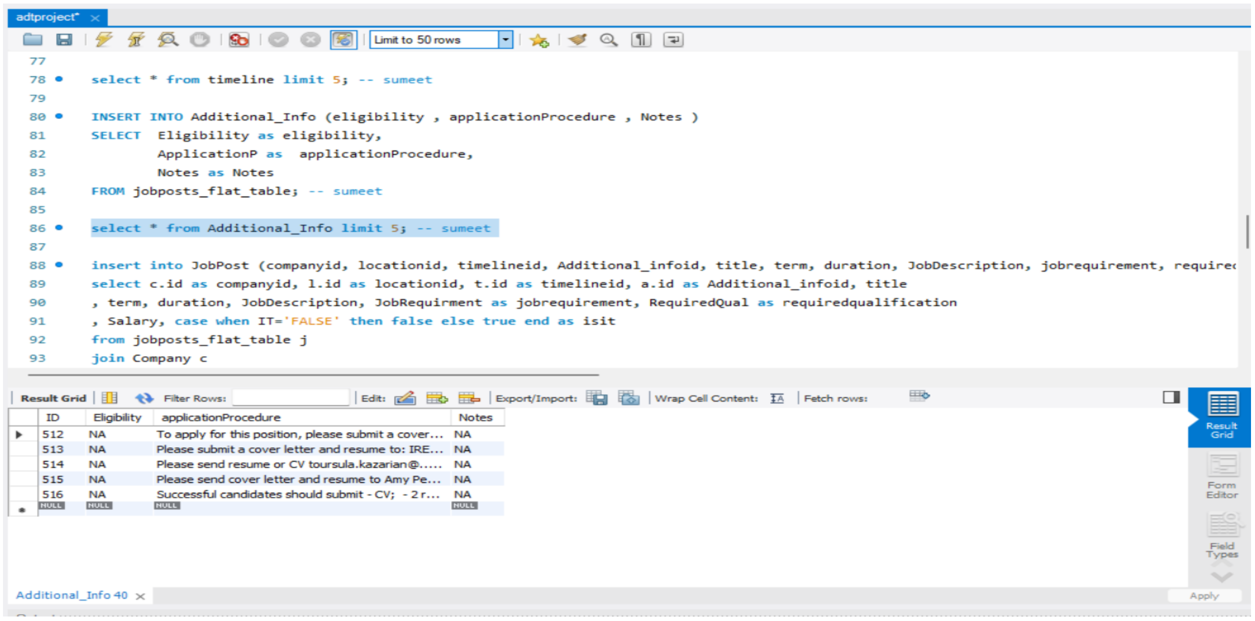
– For timeline table // Sumeet
INSERT INTO timeline(postDate, startDate, deadline, openingDate)
SELECT date as postDate,
StartDate as startDate,
Deadline as deadline,
OpeningDate as openingDate
FROM jobposts_flat_table;



```sql
From jobposts_flat_table
Order by 1;

select * from  location limit 5; -- Nishant


INSERT INTO timeline(postDate, startDate, deadline, openingDate )
SELECT date as postDate,
StartDate as startDate,
Deadline as deadline,
OpeningDate as openingDate
FROM jobposts_flat_table; -- sumeet

select * from timeline limit 5; -- sumeet

INSERT INTO Additional_Info (eligibility , applicationProcedure , Notes )
SELECT  Eligibility as eligibility,
```

ID	postDate	StartDate	deadline	OpeningDate
1869	Jan 5, 2004	NA	26 January 2004	NA
1870	Jan 7, 2004	NA	12 January 2004	NA
1871	Jan 7, 2004	NA	20 January 2004 START DAT...	NA
1872	Jan 7, 2004	NA	23 January 2004 START DAT...	NA
1873	Jan 10, 2004	NA	20 January 2004, 18:00	NA
NULL	NULL	NULL	NULL	NULL

– For Additional_Info table // Sumeet
INSERT INTO Additional_Info (eligibility , applicationProcedure , Notes)
SELECT Eligibility as eligibility,
            ApplicationP as applicationProcedure,
            Notes as Notes
FROM jobposts_flat_table;

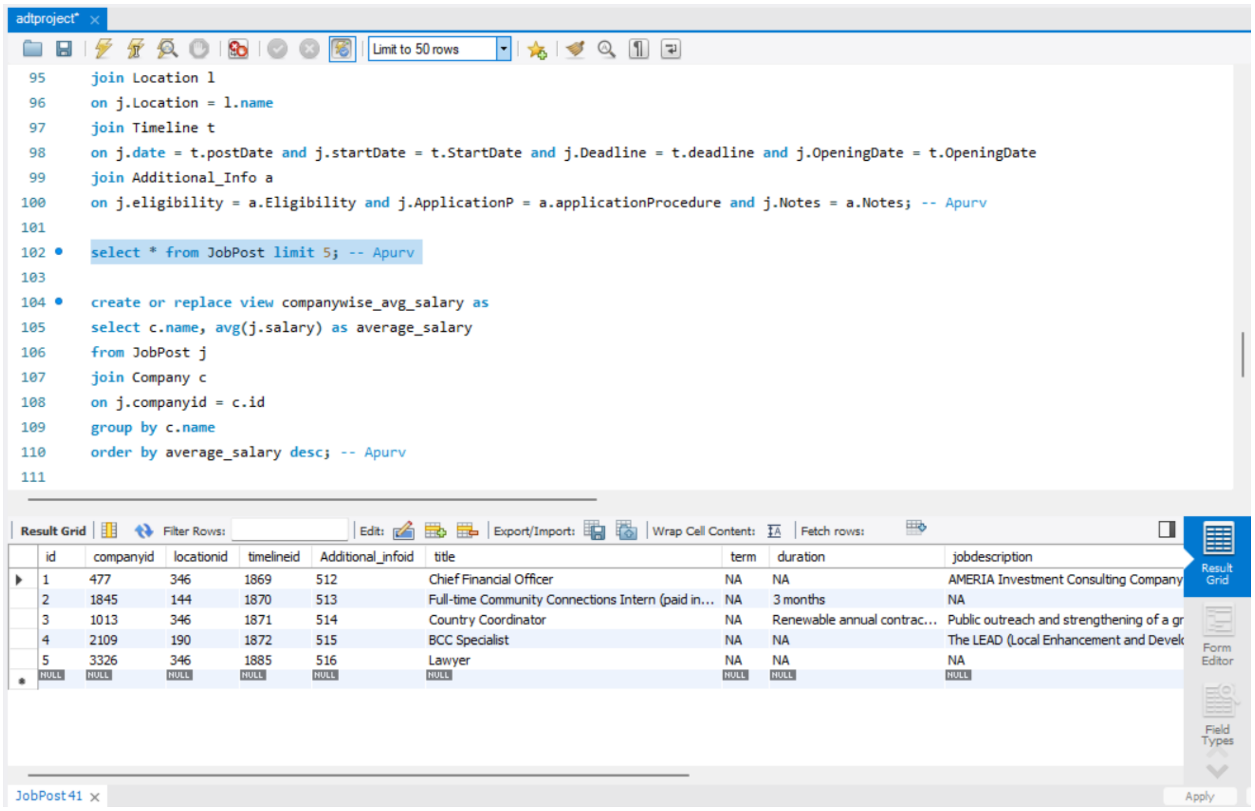


```sql

select * from timeline limit 5; -- sumeet

INSERT INTO Additional_Info (eligibility , applicationProcedure , Notes )
SELECT  Eligibility as eligibility,
        ApplicationP as  applicationProcedure,
        Notes as Notes
FROM jobposts_flat_table; -- sumeet

select * from Additional_Info limit 5; -- sumeet

insert into JobPost (companyid, locationid, timelineid, Additional_infoid, title, term, duration, JobDescription, jobrequirement, requirec
select c.id as companyid, l.id as locationid, t.id as timelineid, a.id as Additional_infoid, title
, term, duration, JobDescription, JobRequirment as jobrequirement, RequiredQual as requiredqualification
, Salary, case when IT='FALSE' then false else true end as isit
from jobposts_flat_table j
join Company c
```

ID	Eligibility	applicationProcedure	Notes
512	NA	To apply for this position, please submit a cover...	NA
513	NA	Please submit a cover letter and resume to: IRE...	NA
514	NA	Please send resume or CV toursula.kazarian@.....	NA
515	NA	Please send cover letter and resume to Amy Pe...	NA
516	NA	Successful candidates should submit - CV; - 2 r...	NA
NULL	NULL	NULL	NULL

– For JobPost table

insert into JobPost (companyid, locationid, timelineid, Additional_infoid, title, term, duration, JobDescription, jobrequirement, requiredqualification,salary, isit)
select c.id as companyid, l.id as locationid, t.id as timelineid, a.id as Additional_infoid, title
, term, duration, JobDescription, JobRequirment as jobrequirement, RequiredQual as requiredqualification
, Salary, case when IT='FALSE' then false else true end as isit
from jobposts_flat_table j
join Company c
on j.Company = c.name
join Location l
on j.Location = l.name
join Timeline t
on j.date = t.postDate and j.startDate = t.StartDate and j.Deadline = t.deadline and j.OpeningDate = t.OpeningDate
join Additional_Info a
on j.eligibility = a.Eligibility and j.ApplicationP = a.applicationProcedure and j.Notes = a.Notes;



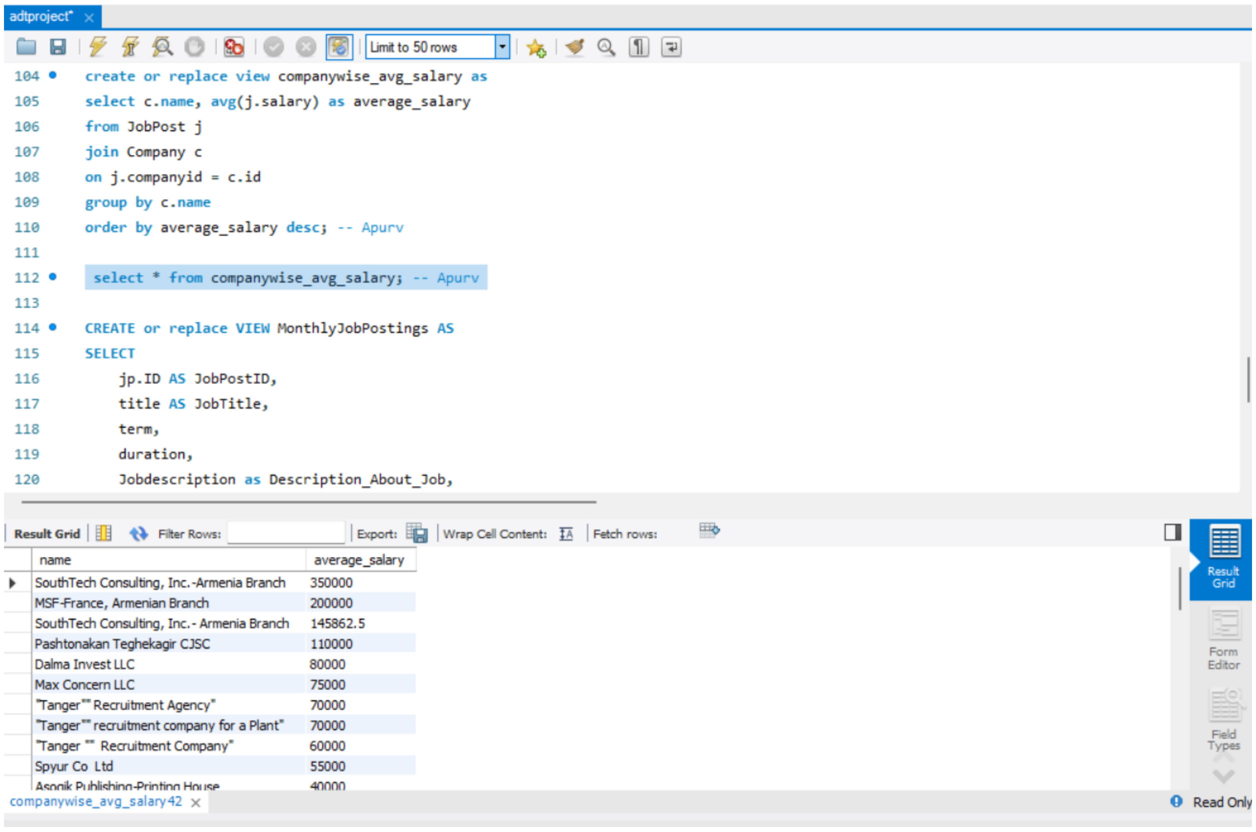
```
join Location l
on j.Location = l.name
join Timeline t
on j.date = t.postDate and j.startDate = t.StartDate and j.Deadline = t.deadline and j.OpeningDate = t.OpeningDate
join Additional_Info a
on j.eligibility = a.Eligibility and j.ApplicationP = a.applicationProcedure and j.Notes = a.Notes; -- Apurv

select * from JobPost limit 5; -- Apurv

create or replace view companywise_avg_salary as
select c.name, avg(j.salary) as average_salary
from JobPost j
join Company c
on j.companyid = c.id
group by c.name
order by average_salary desc; -- Apurv
```

id	companyid	locationid	timelineid	Additional_infoid	title	term	duration	jobdescription
1	477	346	1869	512	Chief Financial Officer	NA	NA	AMERIA Investment Consulting Company
2	1845	144	1870	513	Full-time Community Connections Intern (paid in...	NA	3 months	NA
3	1013	346	1871	514	Country Coordinator	NA	Renewable annual contrac...	Public outreach and strengthening of a gr
4	2109	190	1872	515	BCC Specialist	NA	NA	The LEAD (Local Enhancement and Develo
5	3326	346	1885	516	Lawyer	NA	NA	NA
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Create Views:

Calculating the average salary offered by each company:

create or replace view companywise_avg_salary as
select c.name, avg(j.salary) as average_salary
from JobPost j
join Company c
on j.companyid = c.id
group by c.name
order by average_salary desc; --Apurv

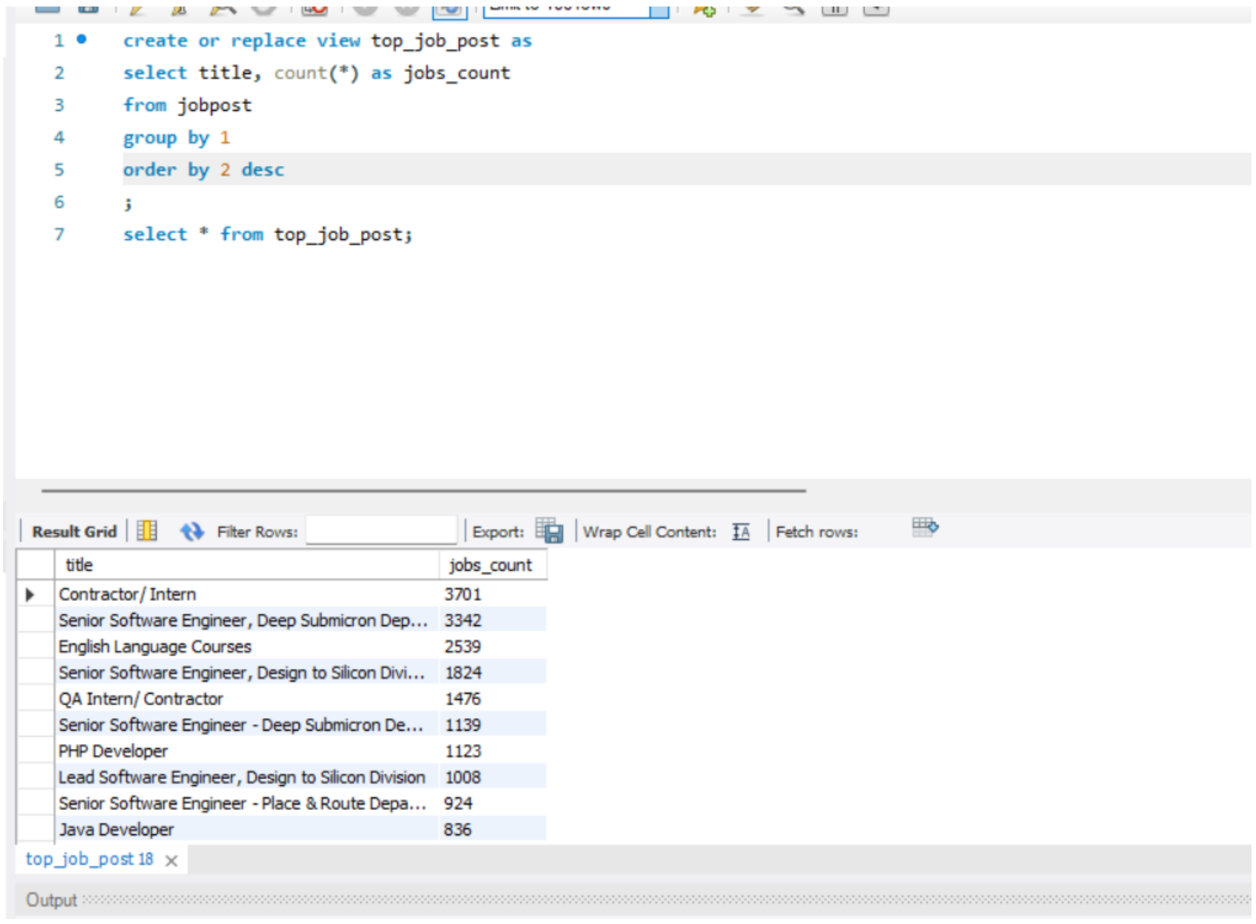


```
create or replace view companywise_avg_salary as
select c.name, avg(j.salary) as average_salary
from JobPost j
join Company c
on j.companyid = c.id
group by c.name
order by average_salary desc; -- Apurv

select * from companywise_avg_salary; -- Apurv

CREATE or replace VIEW MonthlyJobPostings AS
SELECT
    jp.ID AS JobPostID,
    title AS JobTitle,
    term,
    duration,
    Jobdescription as Description_About_Job,
```

name	average_salary
SouthTech Consulting, Inc.-Armenia Branch	350000
MSF-France, Armenian Branch	200000
SouthTech Consulting, Inc.- Armenia Branch	145862.5
Pashtonakan Teghekagir CJSC	110000
Dalma Invest LLC	80000
Max Concern LLC	75000
"Tanger"" Recruitment Agency"	70000
"Tanger"" recruitment company for a Plant"	70000
"Tanger "" Recruitment Company"	60000
Spyur Co Ltd	55000
Asogik Publishing-Printing House	40000

Getting top job title posts:

create or replace view top_job_post as
select title, count(*) as jobs_count
from jobpost
group by 1
order by 2 desc
;
select * from top_job_post; -- Nishant



```
create or replace view top_job_post as
select title, count(*) as jobs_count
from jobpost
group by 1
order by 2 desc
;
select * from top_job_post;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

title	jobs_count
Contractor/ Intern	3701
Senior Software Engineer, Deep Submicron Dep...	3342
English Language Courses	2539
Senior Software Engineer, Design to Silicon Divi...	1824
QA Intern/ Contractor	1476
Senior Software Engineer - Deep Submicron De...	1139
PHP Developer	1123
Lead Software Engineer, Design to Silicon Division	1008
Senior Software Engineer - Place & Route Depa...	924
Java Developer	836

top_job_post 18 ×

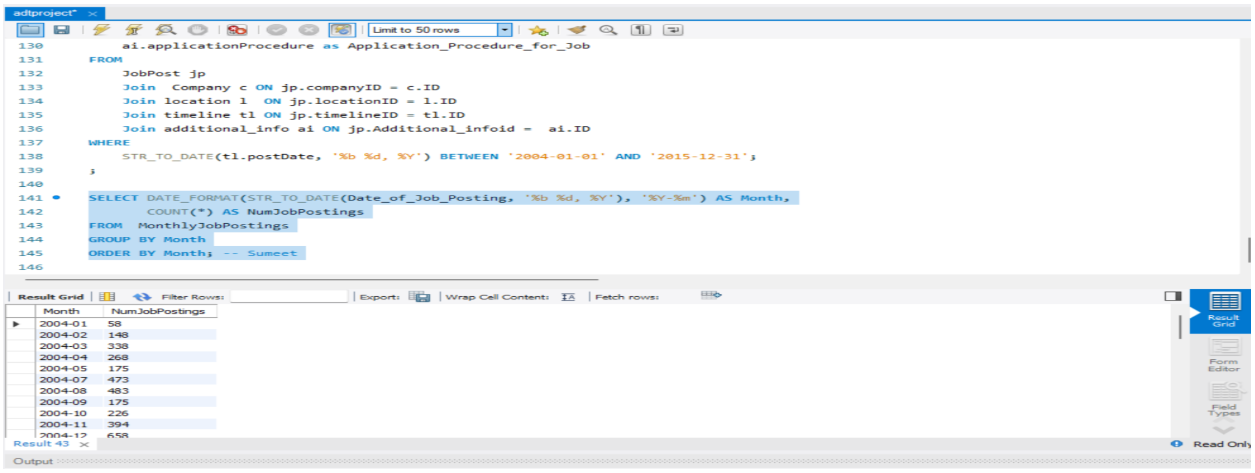
Output

create a view that provides all the information about the job postings that were posted from the time period of 2004 to 2015 – Sumeet

```
–Sumeet
CREATE VIEW MonthlyJobPostings AS
SELECT
    jp.ID AS JobPostID,
    title AS JobTitle,
    term,
    duration,
    Jobdescription as Description_About_Job,
    Jobrequirement as Requirement_About_Job,
    Requiredqualification as Qualification_Required_for_Job,
    salary as Offered_Salary,
    tl.postDate AS Date_of_Job_Posting,
    c.Name as company_Name,
    l.Name as Location_Name,
    tl.startDate as Start_Date_of_Job,
    tl.deadline as Deadline_of_JobApplication,
    ai.eligibility as Eligibity_for_Job,
    ai.applicationProcedure as Application_Procedure_for_Job
FROM
    JobPost jp
    Join  Company c ON jp.companyID = c.ID
    Join location l  ON jp.locationID = l.ID
    Join timeline tl ON jp.timelineID = tl.ID
    Join additional_info ai ON jp.Additional_infoid =  ai.ID
WHERE
    STR_TO_DATE(tl.postDate, '%b %d, %Y') BETWEEN '2004-01-01' AND '2015-12-31';
```

Relevant Question: What is the distribution of job postings per month between 2004 and 2015? –Sumeet

```
SELECT DATE_FORMAT(STR_TO_DATE(Date_of_Job_Posting, '%b %d, %Y'), '%Y-%m') AS
Month,
            COUNT(*) AS NumJobPostings
FROM  MonthlyJobPostings
GROUP BY Month
ORDER BY Month; -- Sumeet
```



```
        ai.applicationProcedure as Application_Procedure_for_Job
    FROM
        JobPost jp
        Join  Company c ON jp.companyID = c.ID
        Join location l  ON jp.locationID = l.ID
        Join timeline tl ON jp.timelineID = tl.ID
        Join additional_info ai ON jp.Additional_infoid =  ai.ID
    WHERE
        STR_TO_DATE(tl.postDate, '%b %d, %Y') BETWEEN '2004-01-01' AND '2015-12-31';
    ;

    SELECT DATE_FORMAT(STR_TO_DATE(Date_of_Job_Posting, '%b %d, %Y'), '%Y-%m') AS Month,
           COUNT(*) AS NumJobPostings
    FROM  MonthlyJobPostings
    GROUP BY Month
    ORDER BY Month; -- Sumeet
```

Month	NumJobPostings
2004-01	58
2004-02	148
2004-03	338
2004-04	268
2004-05	175
2004-07	473
2004-08	483
2004-09	175
2004-10	226
2004-11	394
2004-12	658

Overall Contribution Summary :

Name	Task	Contribution
Nishant	**Conceptional Schema**	Brainstorming of database designing with tables creation and relation mapping.
	Database	Created Company and Location table with constraint and top job title view to get trending job titles
	Code	Created Company and Location table with insert statement from raw data file. Also added foreign key constraints with indexing for query optimization. Also created a view to get trending job titles across the data.
Apurv	**Conceptional Schema**	Brainstorming on dataset to finalize the model
	Database	Designed constraints and keys for jobPost table
	Code	Created a database Developed create and insert statement for Jobpost table. Added suitable data types in create statements as per the data set. Also added foreign key constraints. Created a view to calculate the average salary offered by each company

Sumeet	**Conceptional Schema** **Database** **Code**	Created ER diagram on draw.io and explained the relation types, entities and data types in document Created timeline and Additional_Info table. Also, created constraints for the same. Created timeline and Additional_Info table, defined column data types, database constraints and Insert data statements for timeline and additional_info table. Created a view to get info about distribution of job postings per month between 2004 and 2015

Summary :

This Job Connect - Recruitment Portal database design document is designed to capture the relationships between companies, job locations, timelines, additional information, and individual job postings. The relationships reflect the real-world connections between these entities, providing a comprehensive structure for managing and organizing job-related data within the Online Job posting database. The use of primary and foreign keys ensures data integrity and establishes clear connections between the different tables in our schema. It serves as a valuable guide for understanding the structure and connections within our Job Connect - Recruitment Portal database.