University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 18th October 2023

Group Number: 47

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Abhinav Kansal	24054660	r8u5o@ugrad.cs.ubc.ca	abhinavkansal08@gmail.com
Kashish Joshipura	27745629	t6p2x@ugrad.cs.ubc.ca	kashishjoshipura@gmail.com
Divyansh Sharma	49509540	f9h9g@ugrad.cs.ubc.ca	divyansh.abbott9712@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your email address, and then let us assign you to a TA for your project supervisor.)

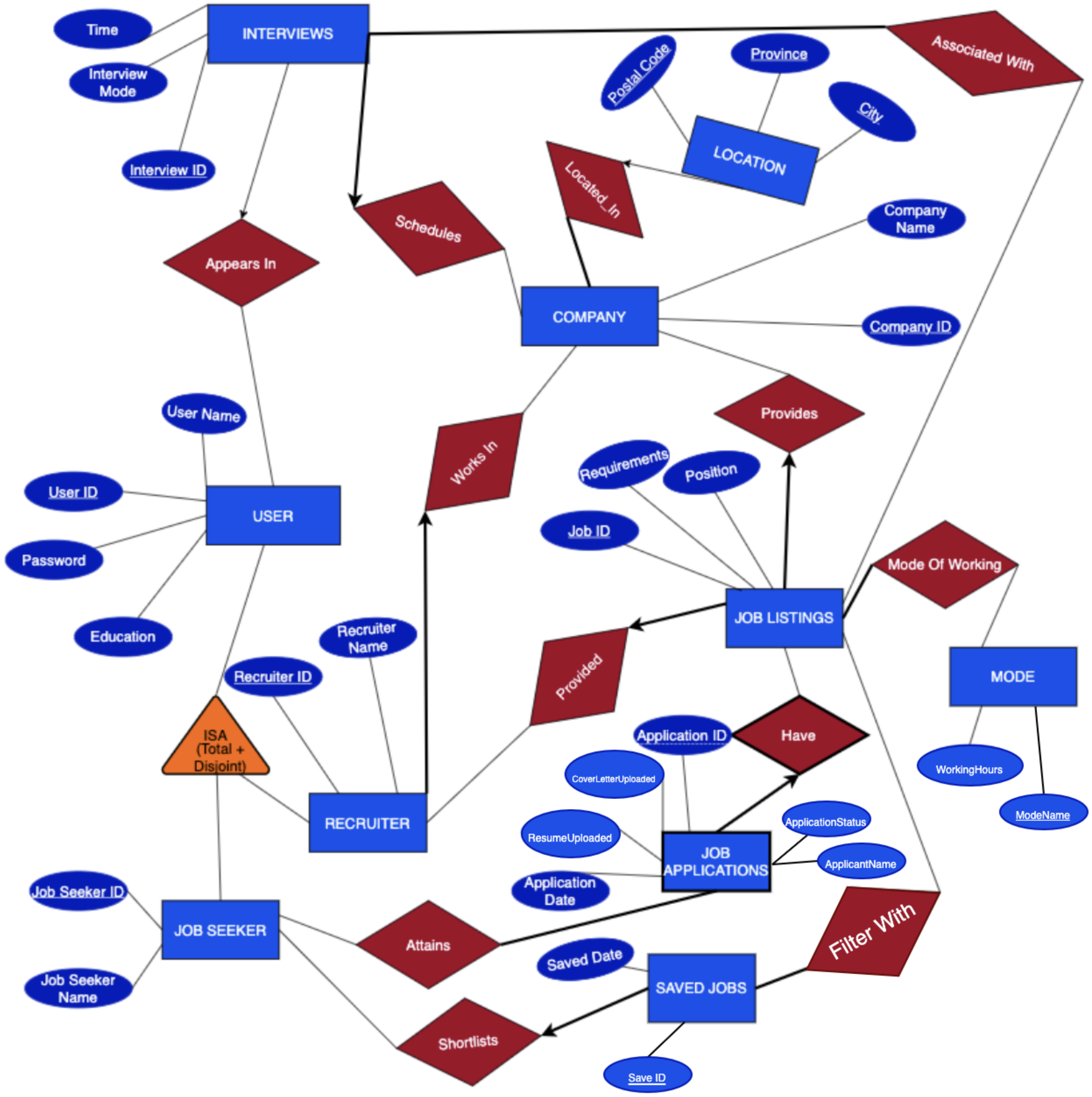
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Deliverables

1. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

The domain for our project will be career, employment, job seeking, and recruiting. Mainly we are going to focus on the job seeking side i.e., from the job seeker's perspective including the functions like applying for jobs, scheduling interviews, shortlisting jobs, and gathering information about the company i.e., where the company is located, and whether the mode of working is remote, hybrid, or in-person. Another side of the project is the ability of the recruiter to directly communicate with job seekers being directly linked with employers and pass any necessary information from employer to employee. The service is that of a mediator.

2. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why. If you have decided not to implement the suggestions given by your project mentor, please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. That being said, your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why in order to better assist the group moving forward.



Changes made to the ER diagram:

1) For Saved Jobs and Mode entities, they don't have a primary key, so we added the primary key by choosing a suitable attribute.
2) For Saved Jobs and Mode entities, they just have one attribute, each entity should have at least one non-key attribute, so we added a suitable complementary attribute so that we can have at least 1 non-key attribute and 1 key-attribute.
3) We added ApplicantName and ApplicationStatus to the JobApplications entity so that we could form valid FDs.

These changes were made so that when we have to lookup on Mode entity and SavedJobs entity, we are effectively selecting tuples from query data and these outputs have unique ids associated so there is no mix up of results.

3. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:

a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.

b. Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

a) List of Table definitions:

Primary Key is represented as underlined

*Foreign Key is represented as **bold***

User(UserID:integer, UserName:varchar, Password:varchar, Education:varchar)

Recruiter(**UserID:integer**, RecruiterName:varchar, **CompanyID:integer**)

JobSeeker(**UserID:integer**, JobSeekerName:varchar)

Company(CompanyID:integer, CompanyName:varchar)

Location(City:varchar, Province:varchar, PostalCode:varchar, Country:varchar)

Interviews(InterviewID:integer, InterviewMode:varchar, **UserID:integer**, **CompanyID:integer**)

JobListing(JobID:integer, Requirements:varchar, Position:varchar, **UserID:integer**, **CompanyID:integer**)

JobApplications(ApplicationID:integer, **JobID:integer**, CoverLetterUploaded:boolean, ResumeUploaded:boolean, ApplicationDate:Date, ApplicantName: varchar, ApplicationStatus: char[8])

SavedJobs(SavedJob#:integer, SavedDate:Date, **UserID:integer**)

Mode(ModeName:varchar, WorkingHours:integer)

Attains(**UserID:integer**, **ApplicationID:integer**, **JobID:integer**)

AssociatedWith(**InterviewID:integer**, **JobID:integer**)

FilterWith(**SaveID:integer**, **JobID:integer**)

b) Specifying the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain

	PRIMARY KEY	**CANDIDATE KEY**	**FOREIGN KEY**
User	UserID	UserID	-
Recruiter	UserID	UserID	UserID, CompanyID NOT NULL
JobSeeker	UserID	UserID	UserID NOT NULL
Company	CompanyID	CompanyID	-
Location	City, Province, PostalCode	City, Province, PostalCode	-
Interviews	InterviewID	InterviewID	UserID, CompanyID NOT NULL
JobListing	JobID	JobID	UserID, CompanyID NOT NULL
JobApplications	ApplicationID, JobID	ApplicationID, JobID	JobID NOT NULL

SavedJobs	SavedJob#	SavedJob#	UserID NOT NULL
Mode	ModeName	ModeName	-
Attains	UserID, ApplicationID, JobID	UserID, ApplicationID, JobId	UserID, ApplicationID, JobID
AssociatedWith	InterviewID, JobID	InterviewID, JobID	InterviewID, JobID
FilterWith	SaveID, JobID	SaveID, JobID	Interview, JobID

4. Functional Dependencies (FDs)

a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key). PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A → A.

Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process.

PK's and CK's FDs :

UserID -> UserName

UserID -> Password

UserID -> Education

–

UserID -> RecruiterName

UserID -> CompanyID

–

UserID -> JobSeekerName

–

CompanyID -> CompanyName

–

City, Province -> PostalCode

PostalCode -> Country

–

InterviewID -> InterviewMode

InterviewID -> UserID

InterviewID -> CompanyID

–

JobID -> Requirements

JobID -> Position

JobID -> UserID

JobID -> CompanyID

JobID -> JobLocation

–

ApplicationID, JobID -> CoverLetterUploaded

ApplicationID, JobID -> ResumeUploaded

ApplicationID, JobID -> ApplicationDate

ApplicationID, JobID -> ApplicationStatus

ApplicationID, JobID -> ApplicantName

–

SavedJob# -> SavedDate

SavedJob# -> UserID

–

ModeName -> WorkingHours

–

SaveID, JobID -> SavedDate

SaveID, JobID -> Requirements

SaveID, JobID -> Position

–

InterviewID, JobID -> Time

InterviewID, JobID -> InterviewMode

InterviewID, JobID -> Requirements

InterviewID, JobID -> Position

Other meaningful FDs:

Requirements -> Position

Eg: Computer Science Degree Completed -> Software Developer

JobID, CoverLetterUploaded -> ApplicationStatus

Eg: 12121, TRUE -> Accepted

JobID, ResumeUploaded -> ApplicantName

Eg: 12124, FALSE -> John Doe

5. Normalization

a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization.

You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post normalization.

- User, already in 3NF.
- Recruiter, already in 3NF.
- JobSeeker, already in 3NF.
- Company, already in 3NF.
- Location, already in 3NF.
- Interview, already in 3NF.
- JobListing, already in 3NF.
- SavedJobs, already in 3NF.
- Mode, already in 3NF.
- Attains, already in 3NF.
- AssociatedWith, already in 3NF.
- FilterWith, already in 3NF.
- JobApplications, not in 3NF so decompose.

Normalization:

Relation: JobApplications(ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicantName, ApplicationStatus)

FDs:

ApplicationID, JobID -> CoverLetterUploaded (1)

ApplicationID, JobID -> ResumeUploaded (2)

ApplicationID, JobID -> ApplicationDate (3)

ApplicationID, JobID -> ApplicationStatus (4)

ApplicationID, JobID -> ApplicantName (5)

JobID, CoverLetterUploaded -> ApplicationStatus (6)

JobID, ResumeUploaded -> ApplicantName (7)

Key: {ApplicationID, JobID}

Closure: {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

JobApplications, not in 3NF because of (6) and (7). LHS of (6) is not a super key and RHS of (6) is not part of the key. LHS of (7) is not a super key and RHS of (7) is not part of the key.

So, we'll use synthesis method decomposition to normalize into 3NF.

Step 1: Find a minimal cover F'

Step 1.1: RHS of FD's should have 1 attribute

All FD's have 1 attribute on RHS.

Step 1.2: Minimize LHS

{ApplicationID}$^+$ = {ApplicationID}

{JobID}$^+$ = {JobID}

(1), (2), (3), (4), and (5) are minimized since JobID doesn’t imply ApplicationID and ApplicationID doesn’t imply JobID.

{CoverLetterUploaded}$^+$ = {CoverLetterUploaded}

{ResumeUploaded}$^+$ = {ResumeUploaded}

(6) and (7) are minimized since JobID doesn’t imply CoverLetterUploaded and CoverLetterUploaded doesn’t imply JobID. JobID doesn’t imply ResumeUploaded and ResumeUploaded doesn’t imply JobID.

Step 1.3: Remove redundant FDs

In (1), ApplicationID, JobID -> CoverLetterUploaded

with (1): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

without (1): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

So, can't remove

In (2), ApplicationID, JobID -> ResumeUploaded

with (2): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

without (2): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

So, can't remove

In (3), ApplicationID, JobID -> ApplicationDate

with (3): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

without (3): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationStatus, ApplicantName}

So, can't remove

In (4), ApplicationID, JobID -> ApplicationStatus

with (4): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

without (4): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

So, we can remove this FD.

In (5), ApplicationID, JobID -> ApplicantName

with (5): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

without (5): {ApplicationID, JobID}$^+$ = {ApplicationID, JobID, CoverLetterUploaded, ResumeUploaded, ApplicationDate, ApplicationStatus, ApplicantName}

So, we can remove this FD.

In (6), JobID, CoverLetterUploaded -> ApplicationStatus

with (6): {JobID, CoverLetterUploaded}$^+$ = {JobID, CoverLetterUploaded, ApplicationStatus}

without (6): {JobID, CoverLetterUploaded}$^+$ = {JobID, CoverLetterUploaded}

So, can't remove

In (7), JobID, ResumeUploaded -> ApplicantName

with (7): {JobID, ResumeUploaded}$^+$ = {JobID, ResumeUploaded, ApplicantName}

without (7): {JobID, ResumeUploaded}$^+$ = {JobID, ResumeUploaded}

So, can't remove

FDs we now have: (1), (2), (3), (6), (7)

Step 2: For each FD X-> b in F' Add relation Xb to the decomposition for R.

R(ApplicationID, JobID,CoverLetterUploaded), **R**(ApplicationID, JobID, ResumeUploaded), **R**(ApplicationID, JobID, ApplicationDate), **R**(JobID, CoverLetterUploaded, ApplicationStatus), **R**(JobID, ResumeUploaded, ApplicantName)

Step 3: If there are no relations in the decomposition that contain all of the attributes of a key, add in a relation that contains all the attributes of a key. This preserves lossless joins.

The set of relations have all attributes of a key.

Final set of relations:

R(ApplicationID, JobID,CoverLetterUploaded), **R**(ApplicationID, JobID, ResumeUploaded), **R**(ApplicationID, JobID, ApplicationDate), **R**(JobID, CoverLetterUploaded, ApplicationStatus), **R**(JobID, ResumeUploaded, ApplicantName)

6. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc. Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use

VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE TABLE User (
UserID        INTEGER,
UserName      VARCHAR,
Password      VARCHAR,
Education     VARCHAR,
PRIMARY KEY(UserID)
)
```

```
CREATE TABLE Recruiter (
RecruiterName     VARCHAR,
UserID            INTEGER,
CompanyID         INTEGER,
PRIMARY KEY(UserID),
FOREIGN KEY(UserID)
```

```
        REFERENCES User(UserID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
FOREIGN KEY(CompanyID)
        REFERENCES Company(CompanyID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
)

CREATE TABLE Recruiter (
RecruiterName       VARCHAR,
UserID              INTEGER,
CompanyID           INTEGER,
PRIMARY KEY (UserID),
FOREIGN KEY(UserID)
        REFERENCES User(UserID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
)

CREATE TABLE JobSeeker (
JobSeekerName       VARCHAR,
UserID              INTEGER,
PRIMARY KEY (UserID),
FOREIGN KEY(UserID)
        REFERENCES User(UserID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
)
```

```
CREATE TABLE Company (
CompanyID      INTEGER,
CompanyName VARCHAR,
PRIMARY KEY(CompanyID)
 )

CREATE TABLE Location (
City             VARCHAR,
Province      VARCHAR,
PostalCode   VARCHAR,
Country        VARCHAR,
PRIMARY KEY (City, Province, PostalCode)
)

CREATE TABLE Interviews (
InterviewID           INTEGER,
InterviewMode       VARCHAR,
UserID                   INTEGER       NOT NULL,
CompanyID             INTEGER       NOT NULL,
PRIMARY KEY(InterviewID),
FOREIGN KEY(UserID)
          REFERENCES User(UserID)
                    DELETE ON CASCADE
                              UPDATE ON CASCADE
FOREIGN KEY(CompanyID)
          REFERENCES Company(CompanyID)
                    DELETE ON CASCADE
                              UPDATE ON CASCADE
```

```
)

CREATE TABLE JobListing (
JobID                   INTEGER,
Requirements            VARCHAR,
Position                VARCHAR,
UserID                  INTEGER     NOT NULL,
CompanyID               INTEGER     NOT NULL,
PRIMARY KEY(JobID),
FOREIGN KEY(UserID)
        REFERENCES User(UserID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
FOREIGN KEY(CompanyID)
        REFERENCES Company(CompanyID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
)

CREATE TABLE JobApplications (
ApplicationID               INTEGER,
JobID                       INTEGER,
CoverLetterUploaded         BOOLEAN,
ResumeUploaded              BOOLEAN,
ApplicationDate     DATE,
ApplicantName       VARCHAR,
ApplicationStatus   CHAR[8],
PRIMARY KEY (ApplicationID, JobID),
FOREIGN KEY(JobID)
```

```
    REFERENCES JobListing(JobID)
        DELETE ON CASCADE
            UPDATE ON CASCADE
)

CREATE TABLE SavedJobs(
SavedJob#           INTEGER,
SavedDate           DATE,
UserID              INTEGER       NOT NULL,
PRIMARY KEY(SavedJob#),
FOREIGN KEY(UserID)
    REFERENCES User(UserID)
        DELETE ON CASCADE
            UPDATE ON CASCADE
)

CREATE TABLE Mode (
ModeName            VARCHAR       PRIMARY KEY,
WorkingHours        INTEGER,
)

CREATE TABLE Attains (
UserID              INTEGER,
ApplicationID       INTEGER,
JobID       INTEGER,
PRIMARY KEY (UserID, ApplicationID, JobID),
FOREIGN KEY(UserID)
    REFERENCES User(UserID)
        DELETE ON CASCADE
```

```
                        UPDATE ON CASCADE
FOREIGN KEY(ApplicationID, JobID)
        REFERENCES JobApplications(ApplicationID)
        REFERENCES JobListing(JobID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
)

CREATE TABLE AssociatedWith (
InterviewID        INTEGER,
JobID              INTEGER,
PRIMARY KEY (InterviewID, JobID),
FOREIGN KEY(InterviewID)
        REFERENCES Interviews(InterviewID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
FOREIGN KEY(JobID)
        REFERENCES JobListing(JobID)
                DELETE ON CASCADE
                        UPDATE ON CASCADE
)

CREATE TABLE FilterWith (
SaveID                  INTEGER,
JobID                   INTEGER,
PRIMARY KEY (SaveID, JobID),
FOREIGN KEY(SaveID)
        REFERENCES SavedJobs(SaveID)
                DELETE ON CASCADE
```

```
            UPDATE ON CASCADE
FOREIGN KEY(JobID)
    REFERENCES JobListing(JobID)
        DELETE ON CASCADE
            UPDATE ON CASCADE
)
```

7. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

```
INSERT INTO User
VALUES(12345, “Adam Troy”, “adam123”, “BSc. in Computer Science”);
INSERT INTO User
VALUES(10201, “Max Tan”, “maxxi”, “BSc. in Statistics”);
INSERT INTO User
VALUES(20190, “Kate Menon”, “km567”, “BSc. in Mathematics”);
INSERT INTO User
VALUES(11111, “Nathon Karl”, “kl1234”, “BSc. in Biology”);
INSERT INTO User
VALUES(90123, “John Miller”, “john312”, “BSc. in Computer Science”);

INSERT INTO Recruiter
VALUES(“Edward Kim”, 22222, 90410);
INSERT INTO Recruiter
VALUES(“Dr. Saint Lauraine”, 88910, 32130);
INSERT INTO Recruiter
VALUES(“Jason Seo”, 78789, 80801);
INSERT INTO Recruiter
VALUES(“Katy Leo”, 31201, 41029);
INSERT INTO Recruiter
```

```
VALUES(“Pamela Ellis”, 33333, 50192);

INSERT INTO JobSeeker
VALUES(“Adam Troy”, 12345);
INSERT INTO JobSeeker
VALUES(“ Max Tan”, 10201);
INSERT INTO JobSeeker
VALUES(“Kate Menon”, 20190);
INSERT INTO JobSeeker
VALUES(“Nathon Karl”, 11111);
INSERT INTO JobSeeker
VALUES(“John Miller”, 90123);

INSERT INTO Company
VALUES(90410, “Apple”);
INSERT INTO Company
VALUES(32130, “Tesla”);
INSERT INTO Company
VALUES(80801, “Microsoft”);
INSERT INTO Company
VALUES(41029, “IBM”);
INSERT INTO Company
VALUES(50192, “Facebook”);

INSERT INTO Location
VALUES(“Toronto”, “Ontario”,  “M5J 0A8”, “Canada”);
INSERT INTO Location
VALUES(“Richmond”, “British Columbia”, “V6X 1S1”, “Canada”);
INSERT INTO Location
```

```
VALUES(“Vancouver”, “British Columbia”, “V6B 1C1”, “Canada”);
INSERT INTO Location
VALUES(“Toronto”, “Ontario”, “M5J 0E7”, “Canada”);
INSERT INTO Location
VALUES(“Toronto”, “Ontario”, “M4P 1E4”, “Canada”);

INSERT INTO Interviews
VALUES(10203, “ Online”, 12345, 90410);
INSERT INTO Interviews
VALUES(10902, “ Online”, 10201, 32130);
INSERT INTO Interviews
VALUES(20304, “ In-Person”, 20190, 80801);
INSERT INTO Interviews
VALUES(40506, “ In-Person”, 11111, 41029);
INSERT INTO Interviews
VALUES(60060, “ Online”, 90123, 50192);

INSERT INTO JobListing
VALUES(55555, “Bsc. in Computer Science”, “Software Developer”,
12345, 90410);
INSERT INTO JobListing
VALUES(54545, “Bsc. in Mathematics”, “Mathematician”, 10201,
 32130);
INSERT INTO JobListing
VALUES(67676, “Bsc. in Computer Science”, “Front End Developer”,
20190, 80801);
INSERT INTO JobListing
VALUES(78651, “Bsc. in Statistics”, “Data Analyst”, 11111, 41029);
INSERT INTO JobListing
```

```
VALUES(30456, “Bsc. in Biology”, “Research Assistant”, 90123, 50192);

INSERT INTO JobApplications
VALUES(10921, 55555, TRUE, TRUE, 2023-09-10, “Adam Troy”, “Accepted”);
INSERT INTO JobApplications
VALUES(20891, 54545, FALSE, TRUE, 2023-09-25, “Max Tan”, “Rejected”);
INSERT INTO JobApplications
VALUES(80801, 67676, TRUE, TRUE, 2023-10-01, “Kate Menon”, “Accepted”);
INSERT INTO JobApplications
VALUES(30121, 78651, FALSE, FALSE, 2023-10-08, “Nathon Karl”, “Pending”);
INSERT INTO JobApplications
VALUES(16080, 30456, TRUE, TRUE, 2023-10-10, “John Miller”, “Accepted”);

INSERT INTO SavedJobs
VALUES(1,  2023-09-01, 12345);
INSERT INTO SavedJobs
VALUES(2, 2023-09-03, 10201);
INSERT INTO SavedJobs
VALUES(3, 2023-09-05, 20190);
INSERT INTO SavedJobs
VALUES(4, 2023-10-05, 11111);
INSERT INTO SavedJobs
VALUES(5, 2023-10-10, 90123);

INSERT INTO Mode
VALUES(“Hybrid”, 35);
INSERT INTO Mode
VALUES(“In-Person”, 35);
INSERT INTO Mode
```

```
VALUES(“Online”, 40);
INSERT INTO Mode
VALUES(“Online”, 40);
INSERT INTO Mode
VALUES(“In-Person”, 40);

INSERT INTO Attains
VALUES(12345, 10921, 55555);
INSERT INTO Attains
VALUES(10201, 20891, 54545);
INSERT INTO Attains
VALUES(20190, 80801, 67676);
INSERT INTO Attains
VALUES(11111, 30121, 78651);
INSERT INTO Attains
VALUES(90123, 16080, 30456);

INSERT INTO AssociatedWith
VALUES(10203, 55555);
INSERT INTO AssociatedWith
VALUES(10902, 54545);
INSERT INTO AssociatedWith
VALUES(20304, 67676);
INSERT INTO AssociatedWith
VALUES(40506, 78651);
INSERT INTO AssociatedWith
VALUES(50192, 30456);

INSERT INTO FilterWith
```

```
VALUES(89102, 55555);
INSERT INTO FilterWith
VALUES(76291, 54545);
INSERT INTO FilterWith
VALUES(90567, 67676);
INSERT INTO FilterWith
VALUES(37498, 78651);
INSERT INTO FilterWith
VALUES(23947, 30456);
```