Module Title	**Database Design and Development**
Assignment Title	**‘Moon’ Job Online Search Platform**
Examination Cycle	**Spring 2023**
Candidate Name	**KHANT NYEIN NAING**
Candidate No	**P00197126**
Centre Name	**KMD INSTITUTE (YANGON)**
Submission Date:	**10 – January - 2023**

Marker's comments:

Moderator's comments:

Mark:	**Moderated Mark:**	**Final Mark:**

Statement and Confirmation of Own Work

Programmed/Qualification name: Level 5 Diploma in Computing

All NCC Education assessed assignments submitted by students must have this statement as the cover page or it will not be accepted for marking. Please ensure that this statement is either firmly attached to the cover of the assignment or electronically inserted into the front of the assignment.

Student declaration

I have read and understood NCC Education's Policy on Academic Dishonesty and Plagiarism.
I can confirm the following details:

Student ID/Registration number	**: P00197126**
Name	**: KHANT NYEIN NAING**
Centre Name	**: KMD Institute (Yangon)**
Module Name	**: Database Design and Development**
Module Leader	**: DAW WAH WAH**
Number of words	**: [2459] words**

I confirm that this is my own work and that I have not plagiarized any part of it. I have also noted the assessment criteria and pass mark for assignments.

Due Date	**: 15 January 2023**
Student Signature	**: KHANT**
Submitted Date	**: 10 January 2023**

Table of Contents

Task – 1 7
1.1 Scenario 7
1.2 Document: 1. Job seekers applying for job 8
1.3 Document: 2. Companies offering jobs 8
1.4 Document: 3. Payments of a job 9
1.5 Document: 4. Interview Result of a job 9
Task – 2 11
(1) Entity Relationship Diagram (ERD) 11
(2) Data Dictionary 13
(2.1) Job Seekers Table 13
(2.2) Companies Table 14
(2.3) Staff Type Table 15
(2.4) Job Location Table 15
(2.5) Job Category Table 16
(2.6) Staff Table 17
(2.8) Job Table 18
(2.9) Interview Table 19
(2.10) Result Table 20
(2.11) Job Monthly Payment Table 21
(2.12) Job Seeker's application details Table 22
Task – 3 24
Normalization 24
Normalization for Document: 1 25
Normalization for Document: 2 27
Normalization for Document: 3 29
Normalization for Document: 4 30
About Anomalies 31
TASK – 4 33
Scripts to create table structures 33
Create table for Jobseekers 33
Create table for Companies 34
Create table for StaffType 34
Create table for JobLocation 35
Create table for JobCategory 36

Create table for Staff 36
Create table for Job 37
Create table for Interviews 38
Create table for Results 39
Create table for Payments 40
Create table for ApplicationDetails 41
Explanation Summary 42
TASK – 5 44
Data population 44
INSERT query and result for Jobseekers 44
INSERT query and result for Companies 45
INSERT query and result for StaffType 46
INSERT query and result for JobLocation 47
INSERT query and result for JobCategory 48
INSERT query and result for Staff 49
INSERT query and result for Job 50
INSERT query and result for Interviews 51
INSERT query and result for Results 52
INSERT query and result for Payments 53
INSERT query and result for ApplicationDetails 54
Explanation Summary 54
TASK –6 56
SQL Reports 56
TASK – 7 62
1.1 Mapped logical database design to physical database design 62
1.1.1 Many to many (Entities to table – 1) 62
1.1.2 One to one (Entities to table -2) 62
1.1.3 One to many (Entities to table-3) 62
1.2 Designed tables for your target DBMS 62
1.3 Derived Data 63
1.3.1 Derived Data (1) 63
1.3.2 Derived Data (2) 63
1.4 Describing about the set of queries that have utility for the business 64
1.5 Writing a report on whether the points outlined in task (1) are met 64
TASK – 8 66

Future Development of a data ware house........ 66
TASK – 9........ 68
Distributed Database Option........ 68
References........ 69
Candidate Checklist........ 71

TASK - 1

Task – 1

“Moon” Job Online Search Platform is a company which is located in Yangon, Myanmar. Before developing this online searched platform was given services as physical. After covid-19, the online order system has developed rapidly and the company's owner decided to develop a job portal online. And the owner assumed that the physical data aka documentation will be stopped from losing as past.

1.1 Scenario

“Moon” is a company which is located in Yangon, Myanmar. They offer a service that accepts job offers and job applications forms online.

The database is needed for this reason. It can be lost when job offer forms and job seekers’ cv forms are stored as physical for many reasons. If these data are stored in database, any person can look for various jobs and can offer jobs for many positions easily.

There are many types of staff in this company. One type of staff can be many staffs. Only the Admin type of staff can control and manage job posts and other data though.

A job seeker can search for many jobs and many job seekers can apply a job. Many jobs can be the same job location aka one job location and many jobs can be the same category aka one category. (For example: To work at the "Bonk" branch company in **`Yangon`** as an IT assistant. In another way, to work at "Yock” main bank in **`Yangon`** as a cyber security engineer.)

When each company post many jobs filling job requirements information like education – fresh graduate in BSc, skills – Java, experience – none, etc. For offering from this online platform, a job will be paid monthly.

A jobseeker may have many interviews and also one job will have many interviews. Each Interview will issue many results because there may be many interview rounds.

1.2 Document: 1. Job seekers applying for job

JobseekerrID	JobseekerName	StaffName	StaffType	JobTitle	Location	JobCategory
JS-001	Michael	Hennery	Admin staff	Web Developer	Singapore	IT
JS-002	John	Van White	Admin manager	Code Tester	Yangon, Myanmar	IT
JS-003	Hazard	Thiago Silver	Admin	IT assistant	Mon, Myanmar	IT
JS-004	Willian	Bourno	Admin staff	IT help desk	Rakhine, Myanmar	IT
JS-005	John	Hennery	Admin staff	Android Developer	Bangkok, Thailand	IT
JS-006	Tony Karoos	Van White	Admin manager	Network Engineer	Yangon, Myanmar	IT
JS-007	Marlin	Thiago Silver	Admin assistant	Accountants	Naypyitaw, Myanmar	Business
JS-008	Messi	Bourno	Admin staff	IT assistant	Mandalay, Myanmar	IT

1.3 Document: 2. Companies offering jobs

CompanyID	CompanyName	JobPosition	JobTitle	JobCategoryName	Location
C-001	Sun	Full-time	Web Developer	IT	Singapore
C-002	Jupiter	Part-time	Code Tester	IT	Yangon, Myanmar
C-003	Butter	Temporary	IT assistant	IT	Mon, Myanmar
C-004	Leaf	Intern	IT help desk	IT	Rakhine, Myanmar
C-005	Super IT	Full-time	Android Developer	IT	Bangkok, Thailand
C-006	Safe IT	Part-time	Network Engineer	IT	Yangon, Myanmar
C-007	Kopa	Temporary	Accountants	Business	Naypyitaw, Myanmar
C-001	Sun	Intern	IT assistant	IT	Mandalay, Myanmar

1.4 Document: 3. Payments of a job

PaymentID	PaymentDate	MonthlyFees	JobTitle	CompanyName
P-001	17-Nov-2022	10$	Code Tester	Sun
P-002	28-Nov-2022	10$	IT assistant	Jupiter
P-003	3-Dec-2022	10$	IT help desk	Butter
P-004	10-Dec-2022	10$	Android Developer	Leaf
P-005	11-Dec-2022	10$	Network Engineer	Super IT
P-006	12-Dec-2022	10$	Accountants	Safe IT
P-007	13-Dec-2022	10$	IT assistant	Kopa
P-008	13-Dec-2022	10$	Web Developer	Sun

1.5 Document: 4. Interview Result of a job

InterviewID	Jobseeker	JobPosition	InterviewedDate	Interview Location	Company Name	InterviewRound	Result
ITR-001	Michael	Web Developer	30-Nov-2022	online	Sun	First round	Pass
ITR-002	John	Code Tester	31-Dec-2022	‘Beat’ Hotel	Jupiter	First round	Pass
ITR-003	Hazard	IT assistant	30-Dec-2022	online	Butter	First round	Pass
ITR-004	Willian	IT help desk	29-Dec-2022	online	Leaf	First round	Pass
ITR-005	John	Android Developer	30-Dec-2022	online	Super IT	Second round	Pass
ITR-006	Tony Karoos	Network Engineer	31-Dec-2022	online	Safe IT	First round	Pass
ITR-007	Marlin	Accountants	30-Dec-2022	online	Kopa	First round	Fail
ITR-008	Messi	IT assistant	28-Dec-2022	online	Sun	First round	Fail

TASK -2

Task – 2

(1) Entity Relationship Diagram (ERD)

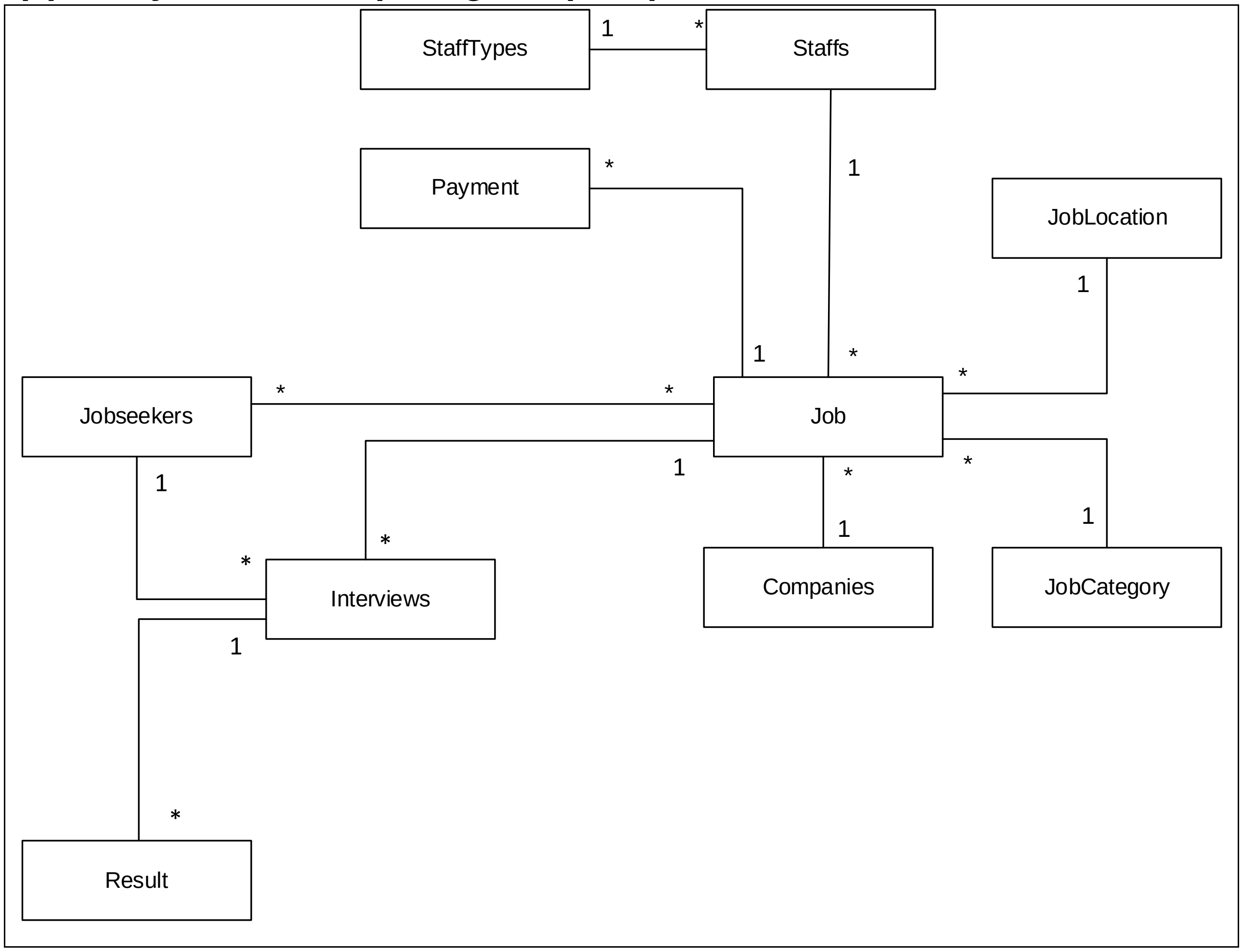


Figure 1 (Entity Relationship Diagram for 'moon' according to scenario)

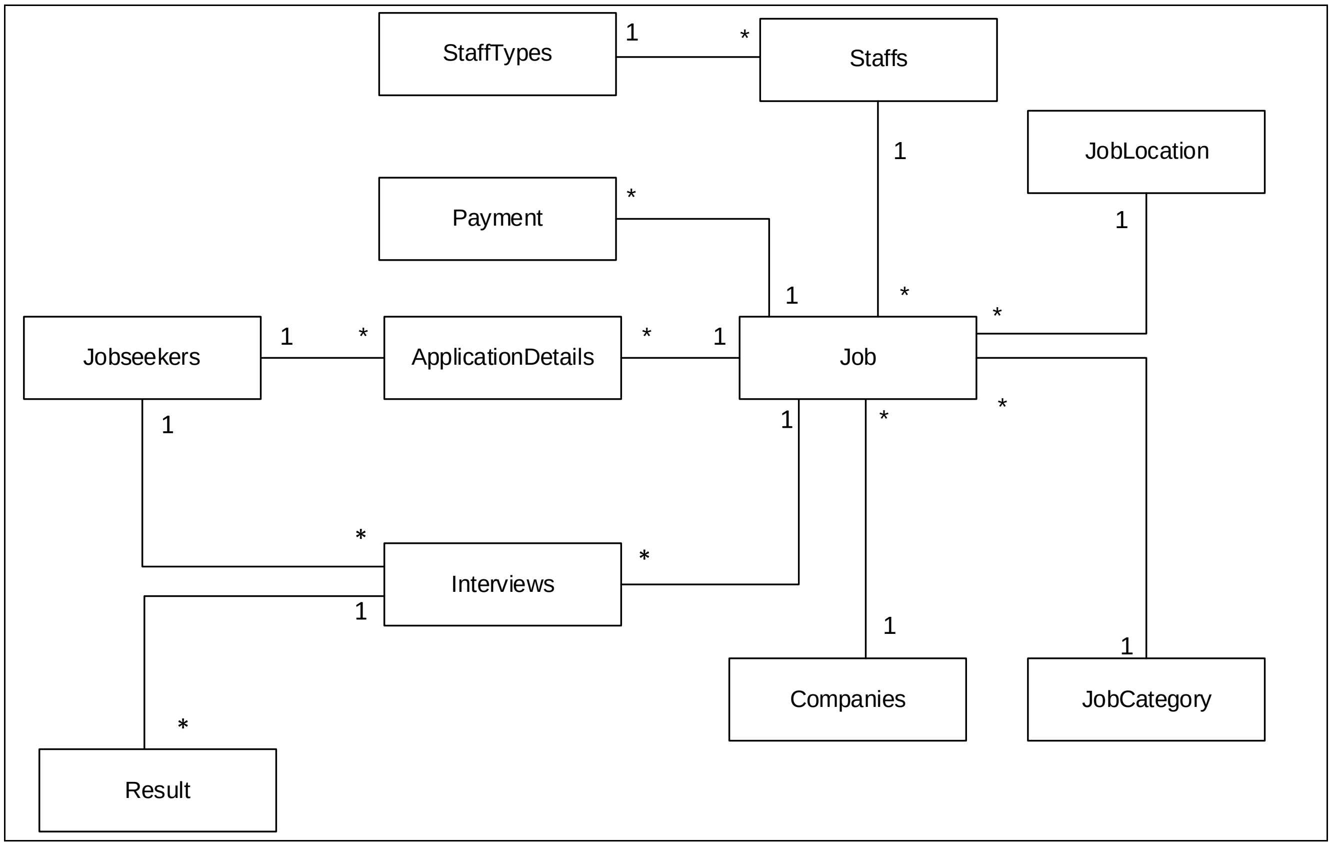


Figure 2 (Update Entity Relationship Diagram for 'moon')

(2) Data Dictionary

(2.1) Job Seekers Table

Entity Name: Jobseekers
Primary Key: JobseekerID
Foreign Key: None

Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
JobseekerID	Varchar	10	It should be started with “JS-” and followed by sequential numbers.	Primary Key, Not null	Unique ID for each job seeker
JobseekerName	Varchar	20		Not null	The name of the job seekers
JobseekerGender	Varchar	10			Gender of the job seekers
JobseekerMail	Varchar	50			Email address of job seekers
JobseekerPhone	Varchar	20			Phone number of job seekers
JobseekerAddress	Varchar	200			Job seekers address
JobseekerSkills	Varchar	200			Job seekers skills
JobseekerExperience	Varchar	200			The experience of job seekers
JobseekerHighEdu	Varchar	200			The highest education of job seekers

(2.2) Companies Table

Entity Name: Companies **Primary Key: CompanyID** **Foreign Key: None**					
Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
CompanyID	Varchar	10	It should be started with “C-” and followed by sequential numbers.	Primary Key, Not Null	Unique ID for each company
CompanyName	Varchar	50		Not null	Company name
CompanyAddress	Varchar	200		Not null	Address of company
CompanyPhone	Varchar	20		Not null	Phone number of company
CompanyEmail	Varchar	50		Not null	Contact email address of company
CompanyWebsite	Varchar	150		Not Null	Company’s website address

(2.3) Staff Type Table

Entity Name: StaffType **Primary Key: StaffTypeID** **Foreign Key: None**					
Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
StaffTypeID	Varchar	10	It should be started with “ST-” and followed by sequential numbers.	Primary Key, Not Null	Unique ID for each Staff Type
StaffType	Varchar	50		Not null	Staff Type name

(2.4) Job Location Table

Entity Name: JobLocation **Primary Key: JobLocationID** **Foreign Key: None**					
Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
JobLocationID	Varchar	10	It should be started with “JL-” and followed by sequential numbers.	Primary Key, Not Null	Unique ID for each job location
RegionState	Varchar	30		Not Null	Job Region or State
country	Varchar	30			Country name of job location

(2.5) Job Category Table

Entity Name: JobCategory **Primary Key: JobCategoryID** **Foreign Key: None**					
Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
JobCategoryID	Varchar	10	It should be started with “JC-” and followed by sequential numbers.	Primary Key	Unique ID for each job category
JobCategoryName	Varchar	50		Not null	The name of job category

(2.6) Staff Table

Entity Name: Staff **Primary Key: StaffID** **Foreign Key: StaffTypeID**					
Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
StaffID	Varchar	10	It should be started with “S-” and followed by sequential numbers.	Primary Key	Unique ID for each staff
StaffName	Varchar	50		Not null	Staff’s name
StaffAddress	Varchar	200		Not null	Address of staff
StaffPhone	Varchar	20		Not null	Phone number of staff
StaffMail	Varchar	50		Not null	Contact email address of staff
StaffTypeID	Varchar	10		Not Null	Id number of staff type

(2.8) Job Table

Entity Name:Jobs

Primary Key: JobID

Foreign Key: JobCategoryID, job_location_it, CompanyID, StaffID

Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
JobID	Varchar	10	It should be started with “JB-” and followed by sequential numbers.	Primary Key	Unique ID for each job
JobTitle	Varchar	100		Not null	Job Title
JobCategoryID	Varchar	10			Job category id number
JobLocationID	Varchar	10			Job location number
CompanyID	Varchar	10		Not null	Job company number
JobPosition	Varchar	50			Job position
JobDescription	text				Job description
JobRequirements	text				Job requirements
JobSalary	decimal				Job salary
NoOfVacancy	int				Number of vacancies
StaffID	Varchar	10			Staff ID number who approved to this post

(2.9) Interview Table

Entity Name:Interviews

Primary Key: InterviewID,

Foreign Key: JobID, JobseekerID

Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
InterviewID	Varchar	10	It should be started with “ITR-” and followed by sequential numbers.	Primary Key, Not Null	Unique ID for each Interview
JobID	Varchar	10		Not Null	Job ID number form Job
JobseekerID	Varchar	10			Jobseeker ID number form Jobseeker
InterviewDate	Date				The date of the interview
InterviewTime	Time				The time of the interview
InterviewLocation	Varchar	100			Interview Location
InterviwerName	Varchar	50			Interviwer Name
TotalInterviewRound	int				Number of Interview Round

(2.10) Result Table

Entity Name:Results Primary Key: ResultID, Foreign Key: InterviewID					
Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
ResultID	Varchar	10	It should be started with “RE-” and followed by sequential numbers.	Primary Key, Not Null	Unique ID for each Result
InterviewID	Varchar	10		Not Null	Job ID number form Job
Result	Varchar	20			Result of each Interview Round
InterviewRound	Varchar	20			Interview Round Name
Reasons	text				Reasons of issuing the result for interview

(2.11) Job Monthly Payment Table

Entity Name: Payment					
Primary Key: PaymentID					
Foreign Key: JobID					
Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
PaymentID	Varchar	10	It should be started with “P-” and followed by sequential numbers.	Primary Key	Unique ID for each job
PaymentDate	date			Not null	The date monthly payment for posting
MonthlyFees	decimal		It will be start with the numbers and end with “$”		Payments fees
JobID	varchar	10			Job ID number

(2.12) Job Seeker's application details Table

Entity Name: ApplicationDetails

Primary Key: JobseekerID+JobID

Foreign Key: JobseekerID, JobID

Attribute Name	**Data Types**	**Size**	**Domain Constraints**	**Integrity Constraints**	**Description**
JobseekerID	Varchar	10	It should be started with "JS-" and followed by sequential numbers.	Primary key, Foreign Key Not null	Unique ID for each job seekers
JobID			It should be started with "JB-" and followed by sequential numbers.		Unique ID for each job
ApplicationStatus	varchar	10			To describe pending, reject, approved

Foreign Key JobseekerID References jobseekers (JobseekerID)
On Cascade Update
On Delete No action
Foreign Key JobID References jobs (JobID)
On Cascade Update
On Delete No action

TASK - 3

Task – 3

Normalization

Normalization helps to clarify and simplify the mixed data from documents or other data forms. It also reduces unnecessary data from these documents and forms breaking down the elements into the parts that cannot be break down anymore and make the data integrity.

This data normalization can be known as database normalization that it is very useful and important part where the relational database design developing because it is beneficial with effective, speed and accuracy.

Tables and columns are clearly appeared when normalizing the data. The data that is related to each other table becomes known clearly. If there is data that is not related to each other, a new table must be split.

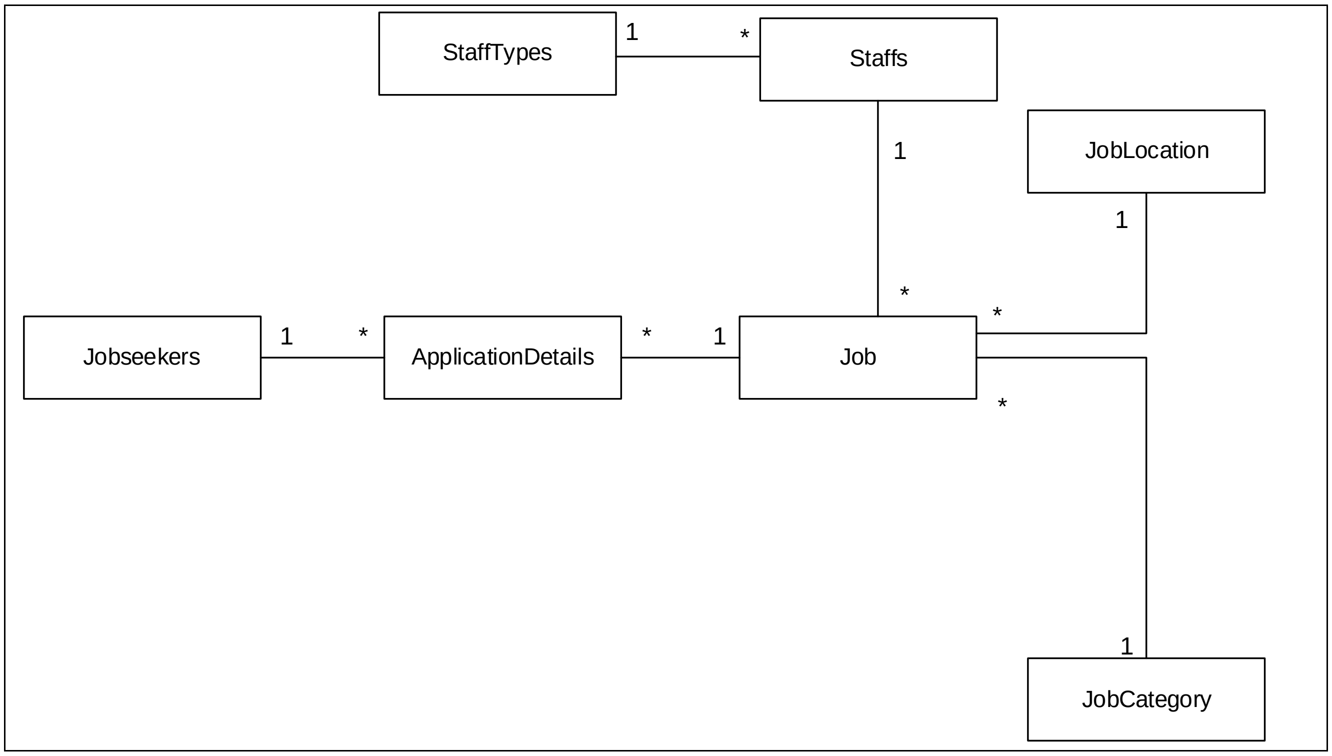
There are five steps in normalization. In first, unnormalized data are grabbed from a sample documents or other sample data form. In second, by analysing these data, divided into single and repeating group. In third, remove the repeating groups. In forth, partial key dependences are split. In fifth, non-key dependences are divided then.

Normalization for Document: 1

UNF	Level	1 NF	2 NF	3 NF	Entities
JobseekerID	1	JobseekerID(PK)	JobseekerID(PK)	JobseekerID(PK)	JobSeekers
jobseekerName	1	jobseekerName	jobseekerName	jobseekerName	
JobseekerGender	1	JobseekerGender	JobseekerGender	JobseekerGender	
JobseekerMail	1	JobseekerMail	JobseekerMail	JobseekerMail	
JobseekerPhone	1	JobseekerPhone	JobseekerPhone	JobseekerPhone	
JobseekerAddress	1	JobseekerAddress	JobseekerAddress	JobseekerAddress	
JobseekerSkills	1	JobseekerSkills	JobseekerSkills	JobseekerSkills	
JobseekerExperience	1	JobseekerExperience	JobseekerExperience	JobseekerExperience	
JobseeekerHighEdu	1	JobseeekerHighEdu	JobseeekerHighEdu	JobseeekerHighEdu	
JobID	2				
JobTitle	2	JobseekerID(FK)	JobseekerID(PK,FK)	JobseekerID(PK,FK)	Application
JobPosition	2	JobID	JobID(PK,FK)	JobID(PK,FK)	Details
JobDescription	2	JobTitle	ApplicationStatus	ApplicationStatus	
JobRequirements	2	JobPosition			Job
JobSalary	2	JobDescription	JobID(PK)	JobID(PK)	
NoOfVacancy	2	JobRequirements	JobTitle	JobTitle	
StaffID	2	JobSalary	JobPosition	JobPosition	
StaffName	2	NoOfVacancy	JobDescription	JobDescription	
StaffAddress	2	StaffID	JobRequirements	JobRequirements	
StaffPhone	2	StaffName	JobSalary	JobSalary	
StaffMail	2	StaffAddress	NoOfVacancy	NoOfVacancy	
StaffTypeID	2	StaffPhone	StaffID	StaffID(FK)	
StaffType	2	StaffMail	StaffName	JobLocationID(FK)	
JobLocationID	2	StaffTypeID	StaffAddress	JobCategoryID(FK)	
RegionState	2	StaffType	StaffPhone		Staffs
Country	2	JobLocationID	StaffMail	StaffID(PK)	
JobCategoryID	2	RegionState	StaffTypeID	StaffName	
JobCategoryName	2	Country	StaffType	StaffAddress	
		JobCategoryID	JobLocationID	StaffPhone	
		JobCategoryName	RegionState	StaffMail	
			Country	StaffTypeID(FK)	
			JobCategoryID		StaffTypes

			JobCategoryName	StaffTypeID(PK) StaffType JobLocationID(PK) RegionState Country JobCategoryID(PK) JobCategoryName	 JobLocation JobCategory

ERD Diagram for Documentation 1

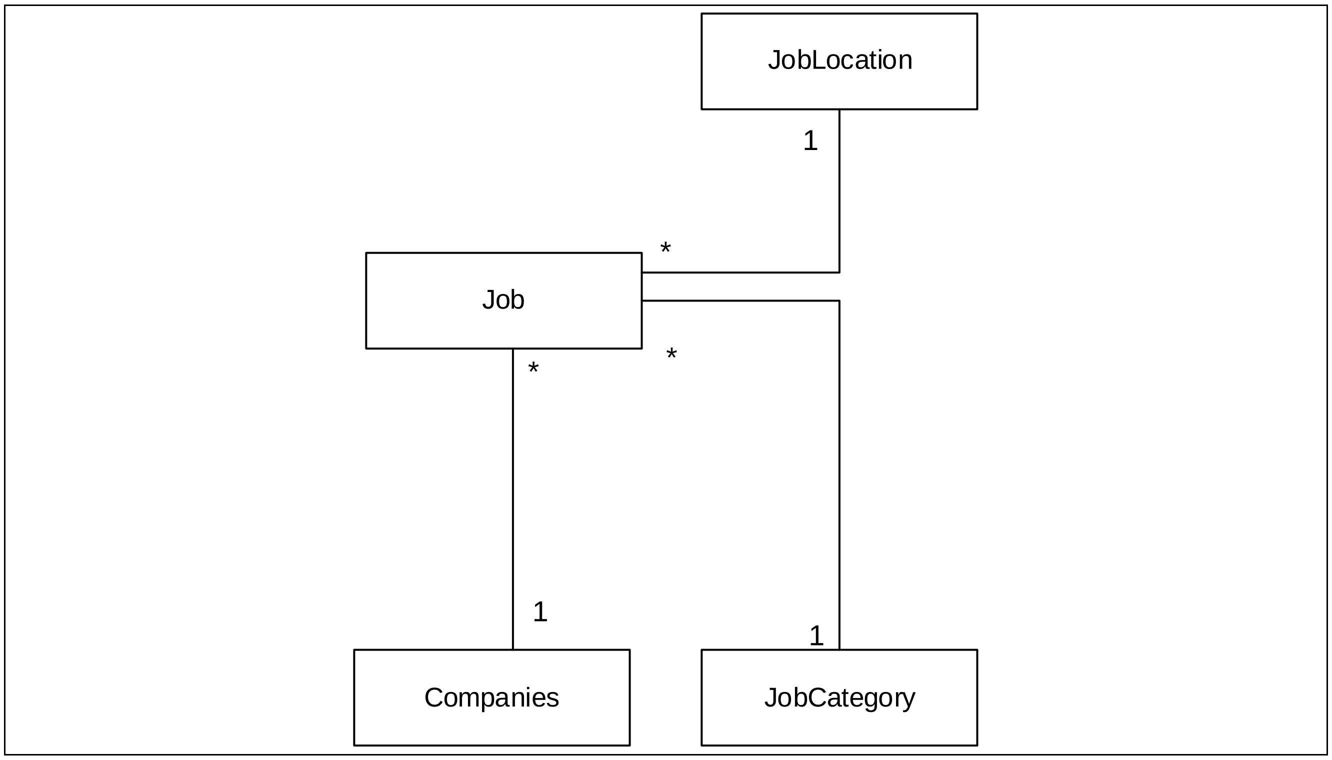


Normalization for Document: 2

UNF	Level	1 NF	2 NF	Entities
CompanyID	1	CompanyID (PK)	CompanyID (PK)	Companies
CompanyName	1	CompanyName	CompanyName	
JobPosition	2	CoampnayAddress	CoampnayAddress	
JobTitle	2	CompanyPhone	CompanyPhone	
JobLocation	2	CompanyEmail	CompanyEmail	
JobCategoryName	2	CompnayWebsite	CompnayWebsite	
		JobID (PK)	JobID (PK)	Job
		CompanyID (FK)	JobTitle	
		JobPosition	CompanyID (FK)	
		JobTitle	JobLocationID (FK)	
		JobLocation	JobCategoryID (FK)	
		JobCategoryName	JobPosition	
			JobDescription	
			JobRequirements	
			JobSalary	
			NoOfVacancy	
			JobLocationID	JobLocation
			RegionState	
			Country	
			JobCategoryID	JobCategory
			JobCategoryName	

Figure 1 (ER Diagram for Document: 2)

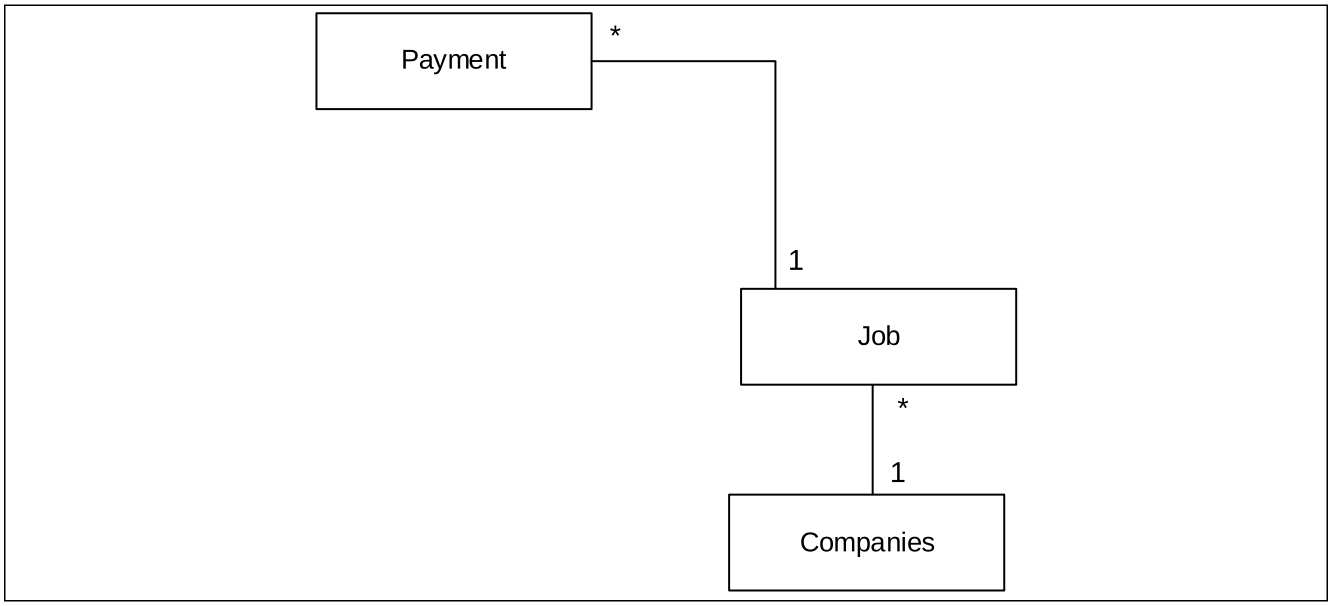
ERD Diagram for Document 2


JobLocation
1
*
Job
*
*
1
1
Companies
JobCategory

Normalization for Document: 3

UNF	Level	1 NF	2 NF	Entities
PaymentID	1	PaymentID (PK)	PaymentID(PK)	Payment
PaymentDate	1	PaymentDate	PaymentDate	
MonthlyFees	1	MonthlyFees JobTitle	MonthlyFees	
JobTitle	2	Company Name	JobID(FK)	
Company	2			
Name	2	JobTitle (PK)	JobID(PK)	Job
		Company Name	JobTitle	
			Company ID(FK)	
			JobPosition	
			JobDescription	
			JobRequirements	
			JobSalary	
			NoOfVacancy	
			CompanyID (PK)	Company
			CompanyName	
			JobPosition	
			JobTitle	
			JobLocation	
			JobCategoryName	

ERD Diagram for Document 3

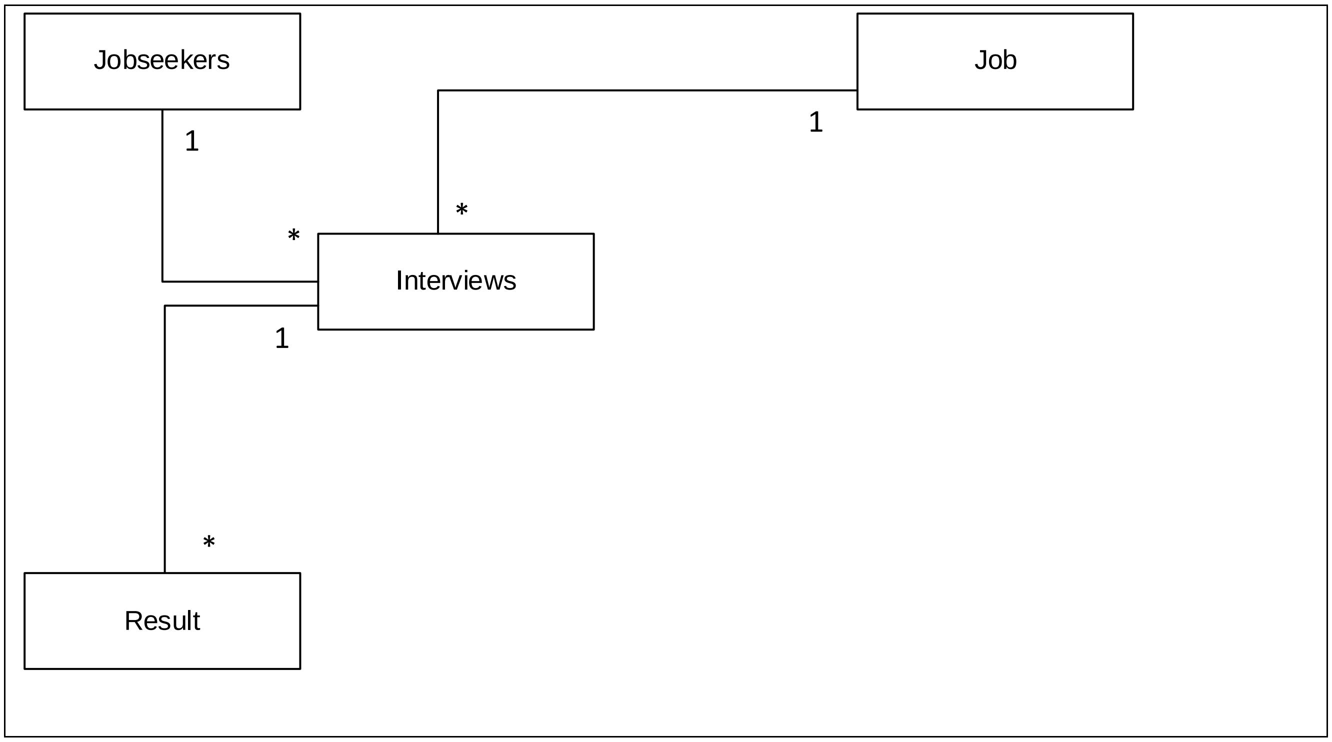


Document: 4. Interview Result of a job

Normalization for Document: 4

UNF	Level	1 NF	2 NF	Entities
InterviewID	2	ResultID (PK)	ResultID (PK)	Results
Jobseeker	2	Result	Result	
JobPosition	2	InterviewRound	InterviewRound	
InterviewedDate	2	Reasons	Reasons	
InterviewLocation	2	InterviewID (FK)	InterviewID(FK)	
CompanyName	2		JobseekerID (FK)	
InterviewRound	1	InterviewID (PK)		
Result	1	InterviewedDate	InterviewID(PK)	Interviews
		InterviewLocation	InterviewDate	
		Jobseeker	InterviewTime	
		JobPosition	InterviewLocation	
		CompanyName	InterviwerName	
		InterviewRound	TotalInterviewRound	
		Result	JobseekerID (FK)	
			JobID (FK)	
			JobseekerID (PK)	Jobseekers
			jobseekerName	
			JobseekerGender	
			JobseekerMail	
			JobseekerPhone	
			JobseekerAddress	
			JobseekerSkills	
			JobseekerExperience	
			JobseeekerHighEdu	
			JobID (PK)	Job
			JobTitle	
			JobPosition	
			JobDescription	
			JobRequirements	
			JobSalary	
			NoOfVacancy	

ERD Diagram for Document 4



About Anomalies

I have used normalization like following to check table are well-structured.

- ➢ Insert Anomalies
- ➢ Update Anomalies
- ➢ Delete Anomalies

❖ **Insert Anomalies**:

We will insert a single company name into document 2. However, if only the company name column is inserted in that document, conflicts may occur because the companyID is unique to the company name, so it cannot be NULL there. After that, the job category set as the primary key, which is another Unique, cannot be omitted.

❖ **Update Anomalies**

Changing the Category of a Job would mean changing it on every activity that was current. For example, JobTitle: IT assistant's JobCategory is IT. If the IT assistant's JobCategory, IT, is changed to another one, business, and updated, every row with that IT assistant must be changed. If there is still little number of data, as fast as it changes, if the data increases, it will become a conflict. Therefore, to overcome this, the tables should be separated.

❖ **Delete Anomalies**

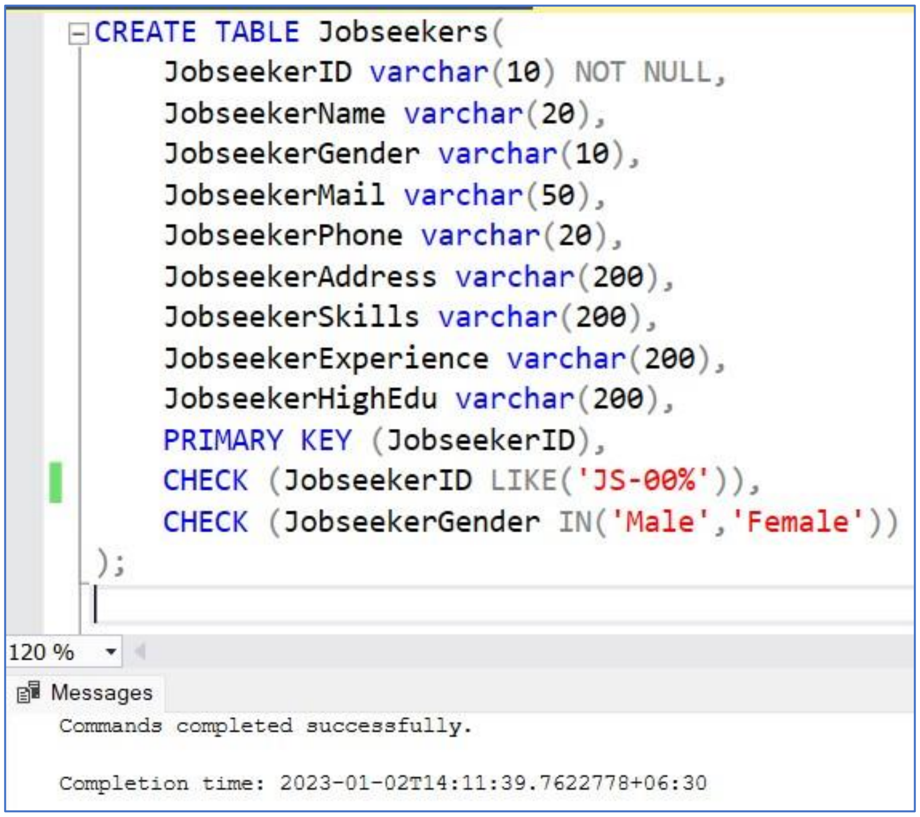
The cause of data loss that we don't want to lose is when one set of data is not properly normalized. If we delete a company ID that is unique in this job offer document such as CompanyID => C-001, any records related to that item will be destroyed and the entire system may fail.

TASK - 4

TASK – 4

Scripts to create table structures

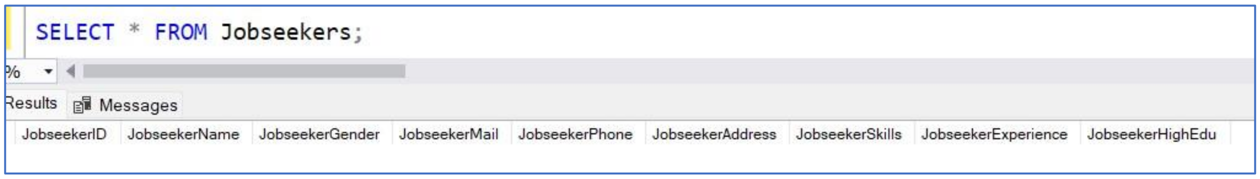
Create table for Jobseekers



```
CREATE TABLE Jobseekers(
    JobseekerID varchar(10) NOT NULL,
    JobseekerName varchar(20),
    JobseekerGender varchar(10),
    JobseekerMail varchar(50),
    JobseekerPhone varchar(20),
    JobseekerAddress varchar(200),
    JobseekerSkills varchar(200),
    JobseekerExperience varchar(200),
    JobseekerHighEdu varchar(200),
    PRIMARY KEY (JobseekerID),
    CHECK (JobseekerID LIKE('JS-00%')),
    CHECK (JobseekerGender IN('Male','Female'))
);
```

```
Commands completed successfully.

Completion time: 2023-01-02T14:11:39.7622778+06:30
```

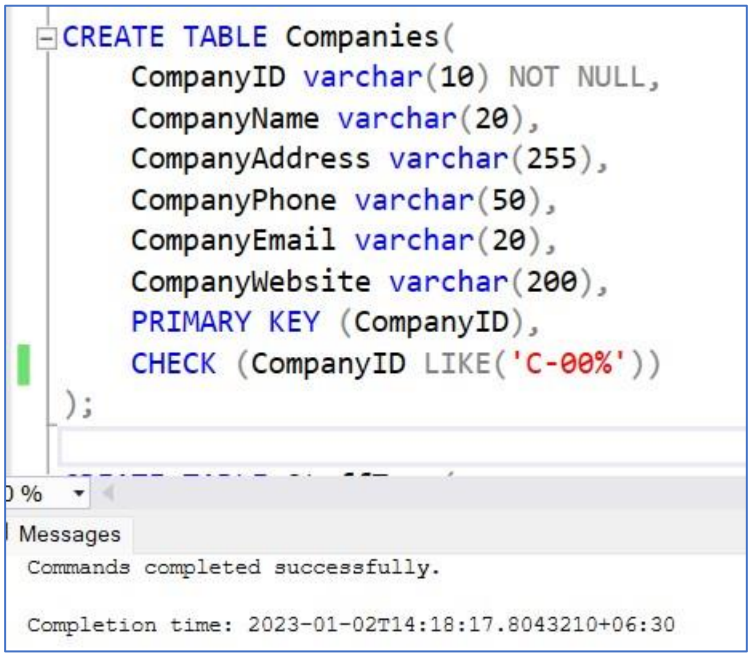
Output of Jobseekers



```
SELECT * FROM Jobseekers;
```

JobseekerID	JobseekerName	JobseekerGender	JobseekerMail	JobseekerPhone	JobseekerAddress	JobseekerSkills	JobseekerExperience	JobseekerHighEdu

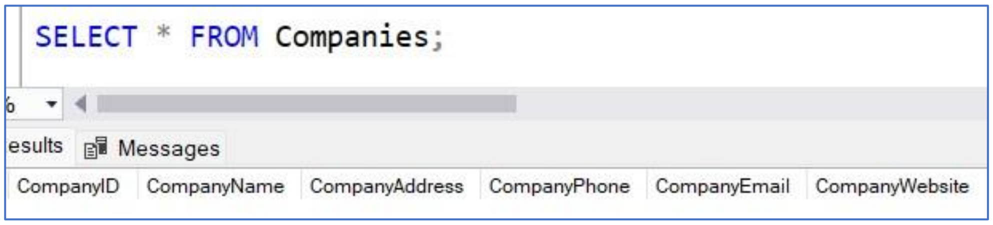
Create table for Companies



```
CREATE TABLE Companies(
    CompanyID varchar(10) NOT NULL,
    CompanyName varchar(20),
    CompanyAddress varchar(255),
    CompanyPhone varchar(50),
    CompanyEmail varchar(20),
    CompanyWebsite varchar(200),
    PRIMARY KEY (CompanyID),
    CHECK (CompanyID LIKE('C-00%'))
);
```

```
Commands completed successfully.

Completion time: 2023-01-02T14:18:17.8043210+06:30
```

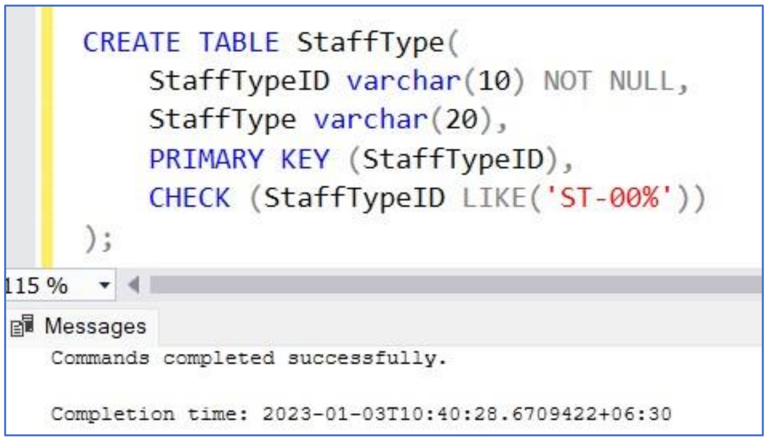
Output of Companies



```
SELECT * FROM Companies;
```

CompanyID	CompanyName	CompanyAddress	CompanyPhone	CompanyEmail	CompanyWebsite

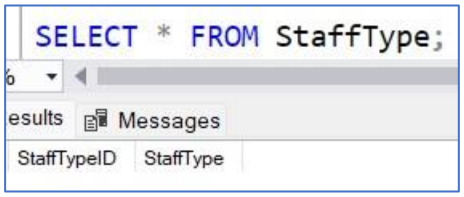
Create table for StaffType



```
CREATE TABLE StaffType(
    StaffTypeID varchar(10) NOT NULL,
    StaffType varchar(20),
    PRIMARY KEY (StaffTypeID),
    CHECK (StaffTypeID LIKE('ST-00%'))
);
```

```
Commands completed successfully.

Completion time: 2023-01-03T10:40:28.6709422+06:30
```

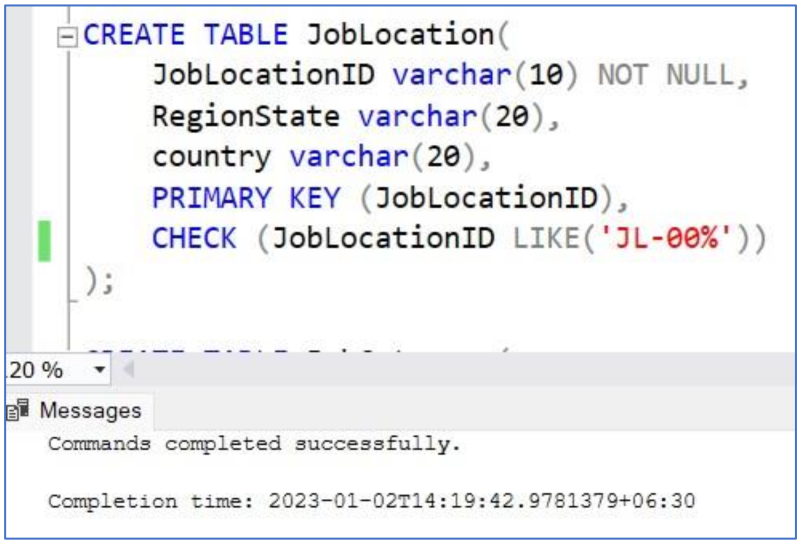
Output of StaffType



```
SELECT * FROM StaffType;
```

```
esults   Messages
StaffTypeID   StaffType
```

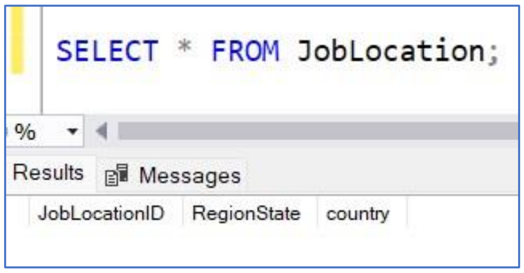
Create table for JobLocation



```
CREATE TABLE JobLocation(
    JobLocationID varchar(10) NOT NULL,
    RegionState varchar(20),
    country varchar(20),
    PRIMARY KEY (JobLocationID),
    CHECK (JobLocationID LIKE('JL-00%'))
);
```

```
20 %
Messages
Commands completed successfully.

Completion time: 2023-01-02T14:19:42.9781379+06:30
```

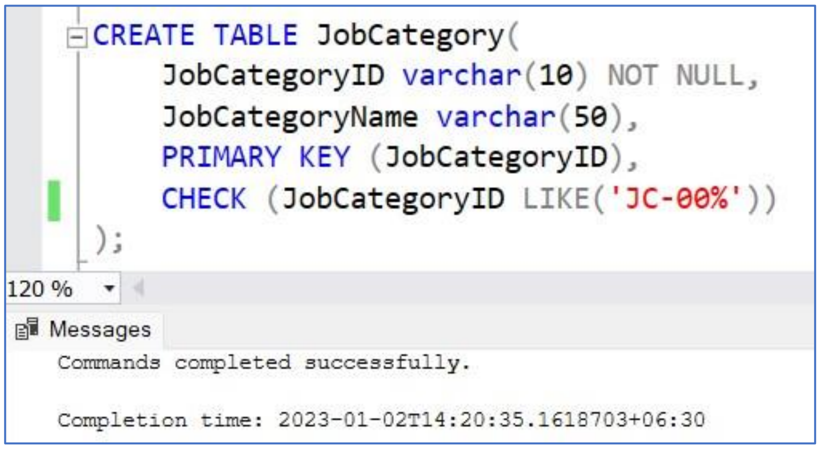
Output of JobLocation



```
SELECT * FROM JobLocation;
```

```
Results   Messages
JobLocationID   RegionState   country
```

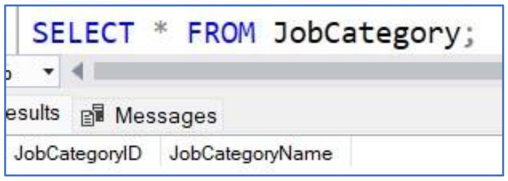
Create table for JobCategory



```
CREATE TABLE JobCategory(
    JobCategoryID varchar(10) NOT NULL,
    JobCategoryName varchar(50),
    PRIMARY KEY (JobCategoryID),
    CHECK (JobCategoryID LIKE('JC-00%'))
);
```

```
Commands completed successfully.

Completion time: 2023-01-02T14:20:35.1618703+06:30
```

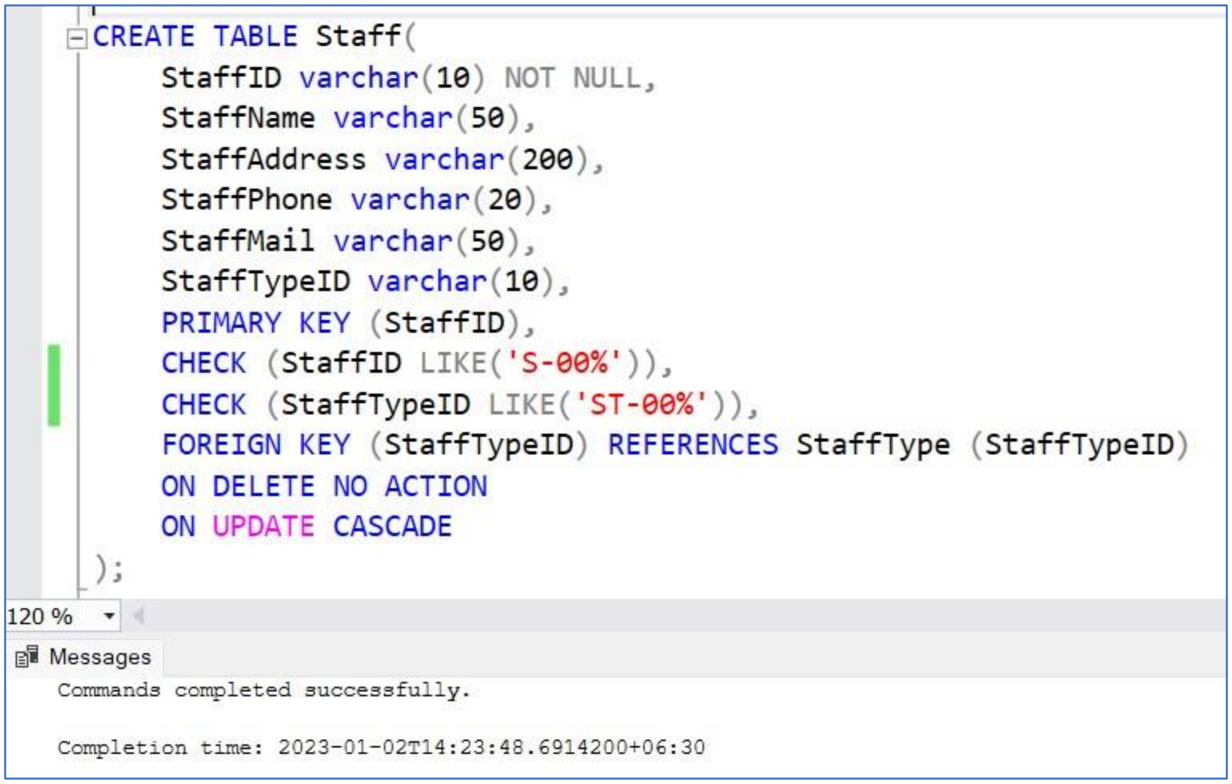
Output of JobCategory



```
SELECT * FROM JobCategory;
```

JobCategoryID	JobCategoryName

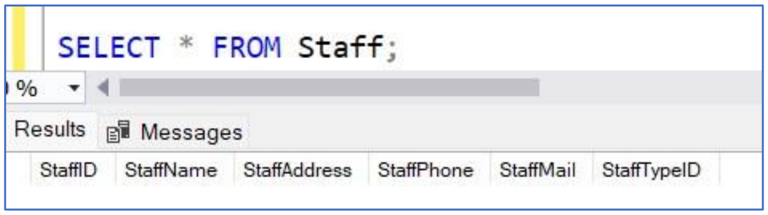
Create table for Staff



```
CREATE TABLE Staff(
    StaffID varchar(10) NOT NULL,
    StaffName varchar(50),
    StaffAddress varchar(200),
    StaffPhone varchar(20),
    StaffMail varchar(50),
    StaffTypeID varchar(10),
    PRIMARY KEY (StaffID),
    CHECK (StaffID LIKE('S-00%')),
    CHECK (StaffTypeID LIKE('ST-00%')),
    FOREIGN KEY (StaffTypeID) REFERENCES StaffType (StaffTypeID)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

```
Commands completed successfully.

Completion time: 2023-01-02T14:23:48.6914200+06:30
```

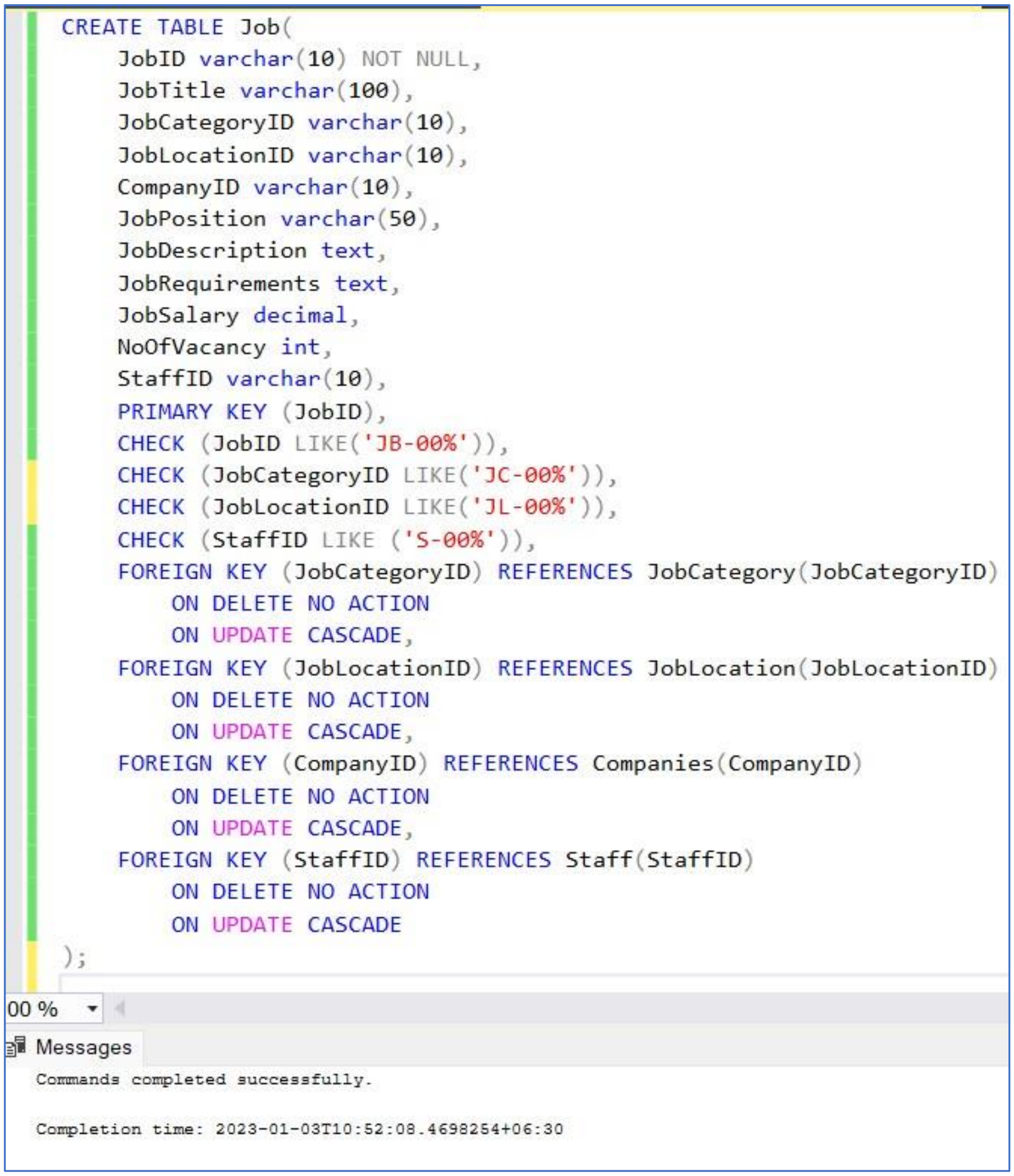
Output of Staff



```
SELECT * FROM Staff;
```

StaffID	StaffName	StaffAddress	StaffPhone	StaffMail	StaffTypeID

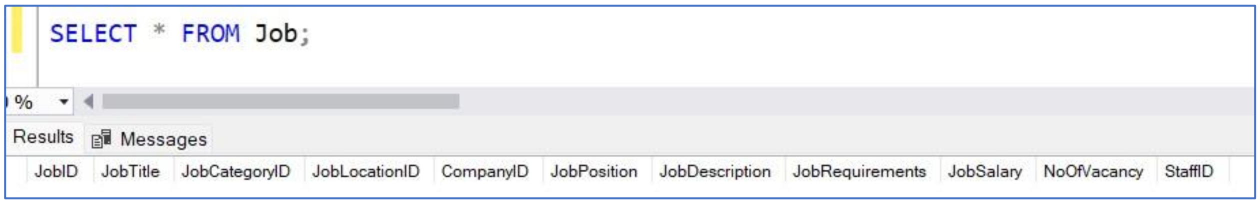
Create table for Job



```
CREATE TABLE Job(
    JobID varchar(10) NOT NULL,
    JobTitle varchar(100),
    JobCategoryID varchar(10),
    JobLocationID varchar(10),
    CompanyID varchar(10),
    JobPosition varchar(50),
    JobDescription text,
    JobRequirements text,
    JobSalary decimal,
    NoOfVacancy int,
    StaffID varchar(10),
    PRIMARY KEY (JobID),
    CHECK (JobID LIKE('JB-00%')),
    CHECK (JobCategoryID LIKE('JC-00%')),
    CHECK (JobLocationID LIKE('JL-00%')),
    CHECK (StaffID LIKE ('S-00%')),
    FOREIGN KEY (JobCategoryID) REFERENCES JobCategory(JobCategoryID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (JobLocationID) REFERENCES JobLocation(JobLocationID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
```

```
Commands completed successfully.

Completion time: 2023-01-03T10:52:08.4698254+06:30
```

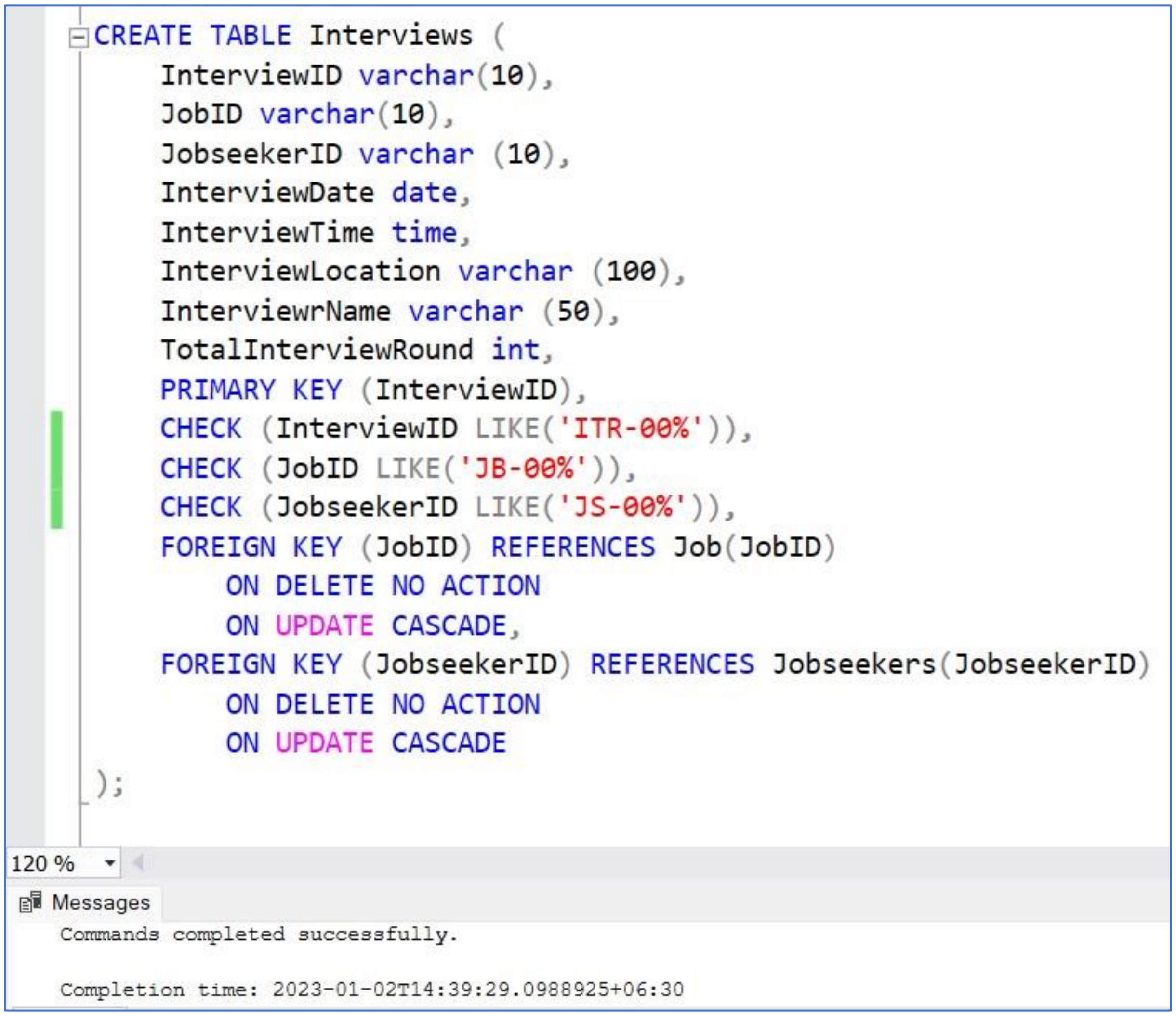
Output of Job



```
SELECT * FROM Job;
```

JobID	JobTitle	JobCategoryID	JobLocationID	CompanyID	JobPosition	JobDescription	JobRequirements	JobSalary	NoOfVacancy	StaffID

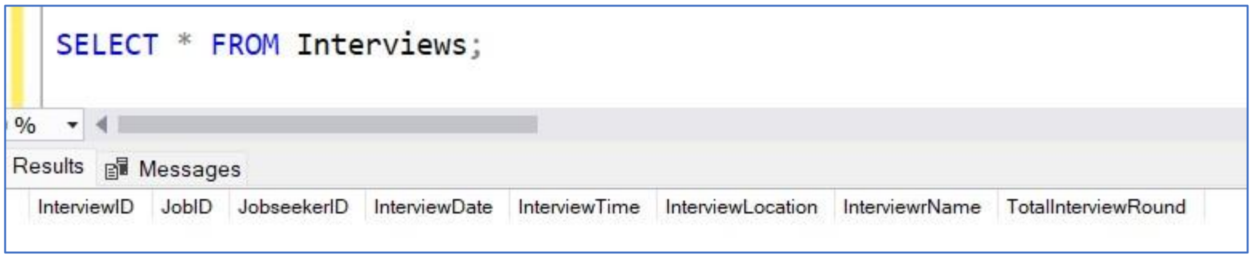
Create table for Interviews



```
CREATE TABLE Interviews (
    InterviewID varchar(10),
    JobID varchar(10),
    JobseekerID varchar (10),
    InterviewDate date,
    InterviewTime time,
    InterviewLocation varchar (100),
    InterviewrName varchar (50),
    TotalInterviewRound int,
    PRIMARY KEY (InterviewID),
    CHECK (InterviewID LIKE('ITR-00%')),
    CHECK (JobID LIKE('JB-00%')),
    CHECK (JobseekerID LIKE('JS-00%')),
    FOREIGN KEY (JobID) REFERENCES Job(JobID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (JobseekerID) REFERENCES Jobseekers(JobseekerID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
```

```
Commands completed successfully.

Completion time: 2023-01-02T14:39:29.0988925+06:30
```

Output of Interviews

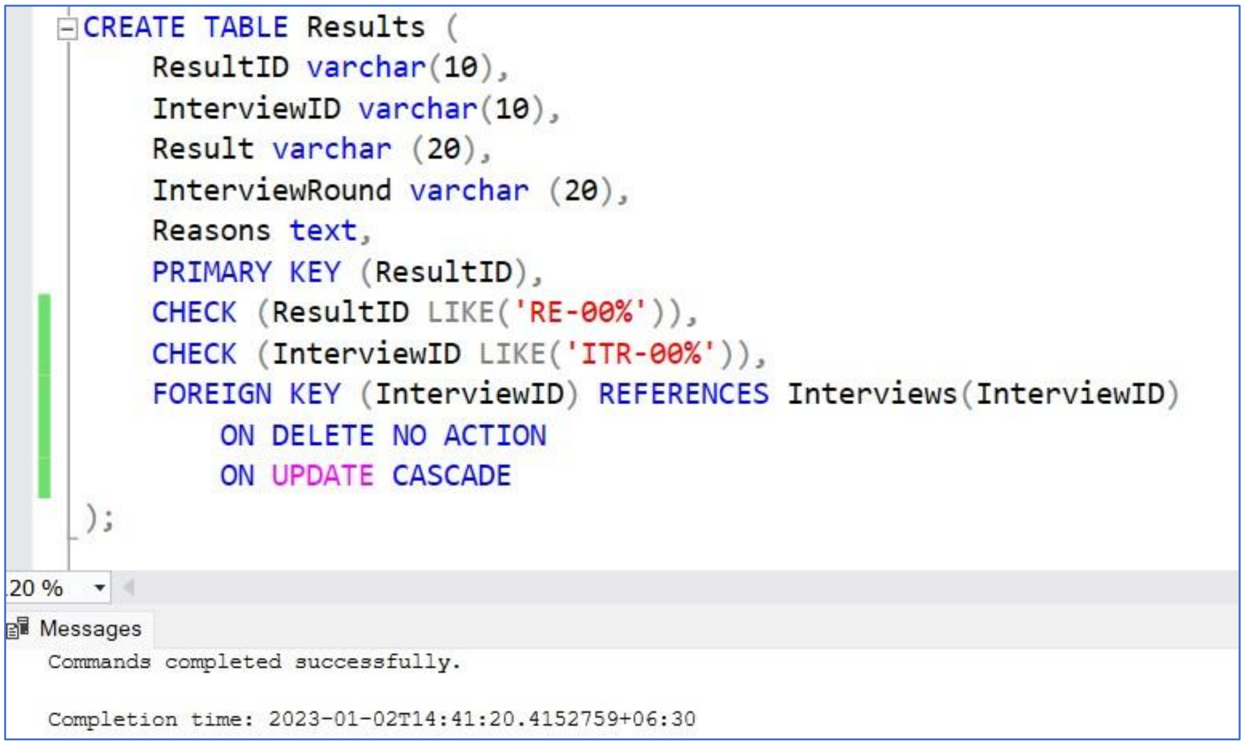


```
SELECT * FROM Interviews;
```

% | Results | Messages

InterviewID	JobID	JobseekerID	InterviewDate	InterviewTime	InterviewLocation	InterviewrName	TotalInterviewRound

Create table for Results

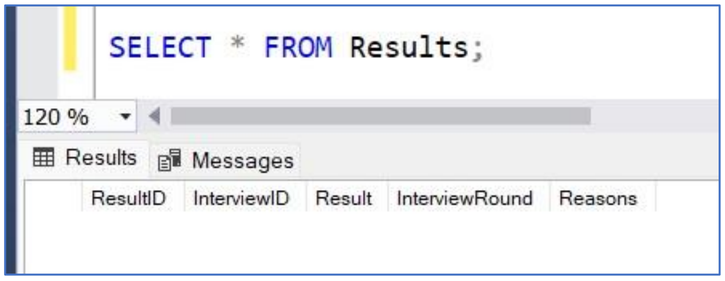


```
CREATE TABLE Results (
    ResultID varchar(10),
    InterviewID varchar(10),
    Result varchar (20),
    InterviewRound varchar (20),
    Reasons text,
    PRIMARY KEY (ResultID),
    CHECK (ResultID LIKE('RE-00%')),
    CHECK (InterviewID LIKE('ITR-00%')),
    FOREIGN KEY (InterviewID) REFERENCES Interviews(InterviewID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
```

20 % | Messages

```
Commands completed successfully.

Completion time: 2023-01-02T14:41:20.4152759+06:30
```

Output of Results

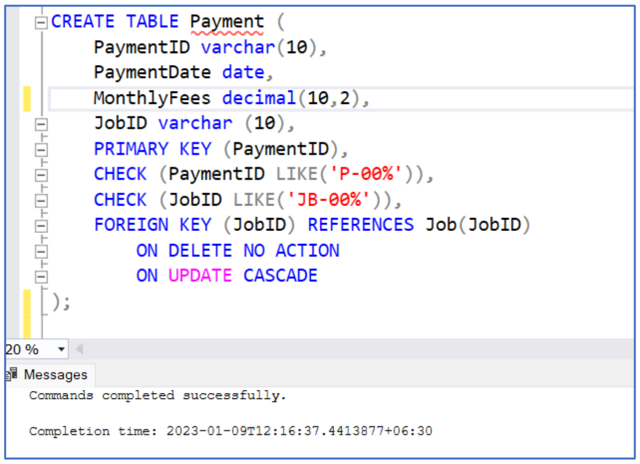


```
SELECT * FROM Results;
```

120 % | Results | Messages

ResultID	InterviewID	Result	InterviewRound	Reasons

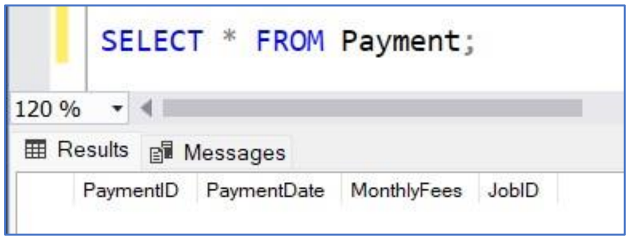
Create table for Payments



```
CREATE TABLE Payment (
    PaymentID varchar(10),
    PaymentDate date,
    MonthlyFees decimal(10,2),
    JobID varchar (10),
    PRIMARY KEY (PaymentID),
    CHECK (PaymentID LIKE('P-00%')),
    CHECK (JobID LIKE('JB-00%')),
    FOREIGN KEY (JobID) REFERENCES Job(JobID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
```

```
Commands completed successfully.

Completion time: 2023-01-09T12:16:37.4413877+06:30
```

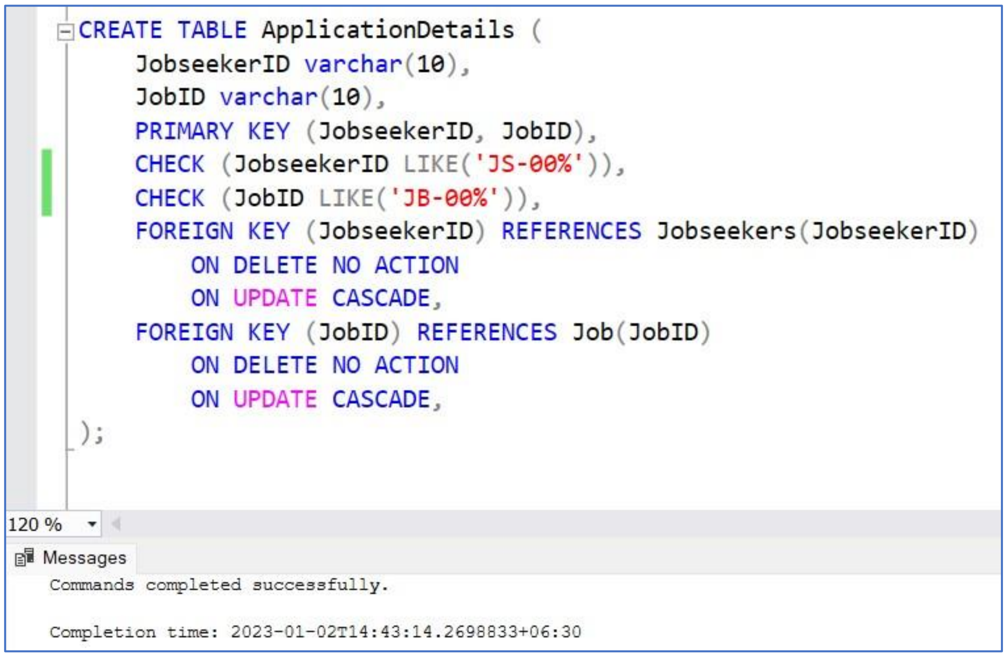
Output of Payments



```
SELECT * FROM Payment;
```

PaymentID	PaymentDate	MonthlyFees	JobID

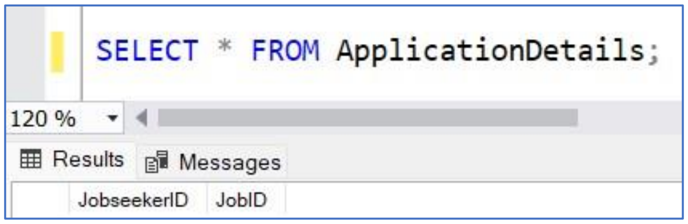
Create table for ApplicationDetails



```
CREATE TABLE ApplicationDetails (
    JobseekerID varchar(10),
    JobID varchar(10),
    PRIMARY KEY (JobseekerID, JobID),
    CHECK (JobseekerID LIKE('JS-00%')),
    CHECK (JobID LIKE('JB-00%')),
    FOREIGN KEY (JobseekerID) REFERENCES Jobseekers(JobseekerID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
    FOREIGN KEY (JobID) REFERENCES Job(JobID)
        ON DELETE NO ACTION
        ON UPDATE CASCADE,
);
```

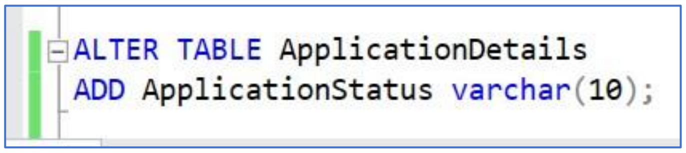
```
Commands completed successfully.

Completion time: 2023-01-02T14:43:14.2698833+06:30
```

Output of ApplicationDetails



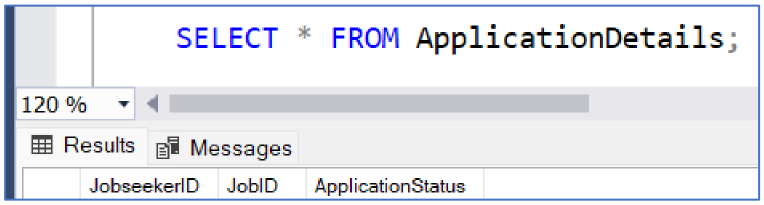
```
SELECT * FROM ApplicationDetails;
```

	JobseekerID	JobID



```
ALTER TABLE ApplicationDetails
ADD ApplicationStatus varchar(10);
```

Output for alter table



```
SELECT * FROM ApplicationDetails;
```

	JobseekerID	JobID	ApplicationStatus

Explanation Summary

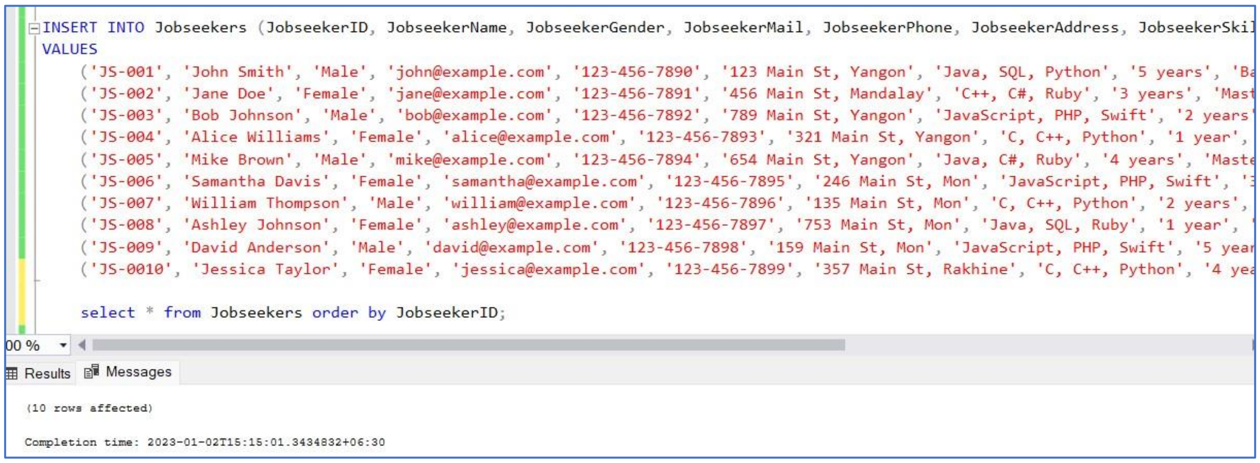
In creating the database, the SQL Create statement was used to build the tables. It was important to follow a logical order, first creating standalone tables and then building tables that were connected to them. Dummy tables were created last. One issue encountered was using the wrong table name, which required deleting any dependencies before the table could be removed. Tasks 2 and 3 required careful planning to maintain a compact table structure. Careful consideration was given to selecting the appropriate data types, including using decimal or integer for currency, text for large numbers, date and time for dates and times, and varchar receive any strings.

TASK - 5

TASK – 5

Data population

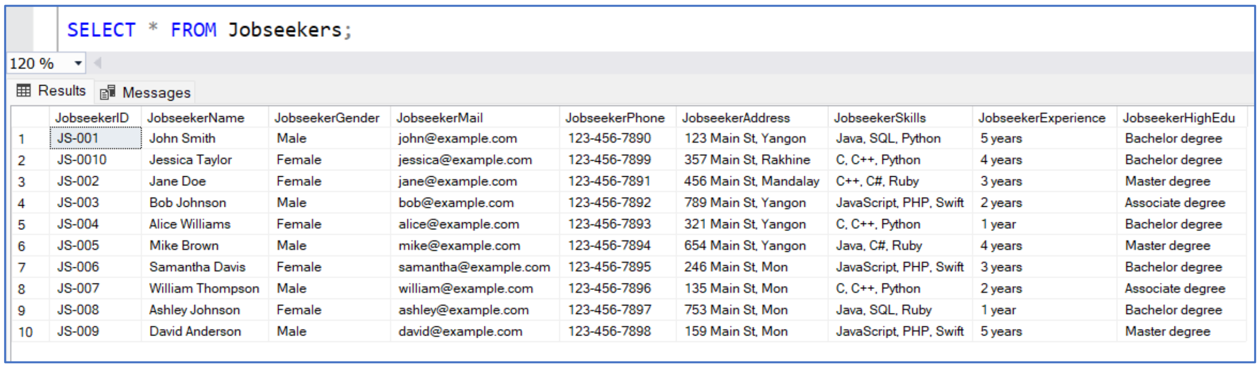
INSERT query and result for Jobseekers



```
INSERT INTO Jobseekers (JobseekerID, JobseekerName, JobseekerGender, JobseekerMail, JobseekerPhone, JobseekerAddress, JobseekerSkil
VALUES
    ('JS-001', 'John Smith', 'Male', 'john@example.com', '123-456-7890', '123 Main St, Yangon', 'Java, SQL, Python', '5 years', 'Ba
    ('JS-002', 'Jane Doe', 'Female', 'jane@example.com', '123-456-7891', '456 Main St, Mandalay', 'C++, C#, Ruby', '3 years', 'Mast
    ('JS-003', 'Bob Johnson', 'Male', 'bob@example.com', '123-456-7892', '789 Main St, Yangon', 'JavaScript, PHP, Swift', '2 years'
    ('JS-004', 'Alice Williams', 'Female', 'alice@example.com', '123-456-7893', '321 Main St, Yangon', 'C, C++, Python', '1 year',
    ('JS-005', 'Mike Brown', 'Male', 'mike@example.com', '123-456-7894', '654 Main St, Yangon', 'Java, C#, Ruby', '4 years', 'Maste
    ('JS-006', 'Samantha Davis', 'Female', 'samantha@example.com', '123-456-7895', '246 Main St, Mon', 'JavaScript, PHP, Swift', '3
    ('JS-007', 'William Thompson', 'Male', 'william@example.com', '123-456-7896', '135 Main St, Mon', 'C, C++, Python', '2 years',
    ('JS-008', 'Ashley Johnson', 'Female', 'ashley@example.com', '123-456-7897', '753 Main St, Mon', 'Java, SQL, Ruby', '1 year',
    ('JS-009', 'David Anderson', 'Male', 'david@example.com', '123-456-7898', '159 Main St, Mon', 'JavaScript, PHP, Swift', '5 year
    ('JS-0010', 'Jessica Taylor', 'Female', 'jessica@example.com', '123-456-7899', '357 Main St, Rakhine', 'C, C++, Python', '4 yea

    select * from Jobseekers order by JobseekerID;
```

```
(10 rows affected)

Completion time: 2023-01-02T15:15:01.3434832+06:30
```

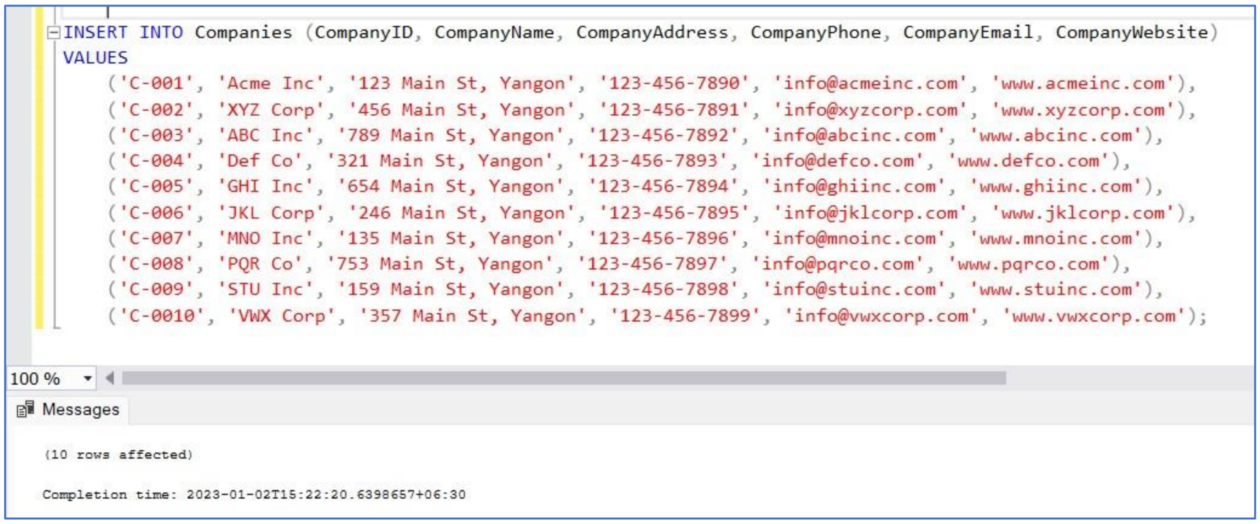
Output for Jobseekers



```
SELECT * FROM Jobseekers;
```

	JobseekerID	JobseekerName	JobseekerGender	JobseekerMail	JobseekerPhone	JobseekerAddress	JobseekerSkills	JobseekerExperience	JobseekerHighEdu
1	JS-001	John Smith	Male	john@example.com	123-456-7890	123 Main St, Yangon	Java, SQL, Python	5 years	Bachelor degree
2	JS-0010	Jessica Taylor	Female	jessica@example.com	123-456-7899	357 Main St, Rakhine	C, C++, Python	4 years	Bachelor degree
3	JS-002	Jane Doe	Female	jane@example.com	123-456-7891	456 Main St, Mandalay	C++, C#, Ruby	3 years	Master degree
4	JS-003	Bob Johnson	Male	bob@example.com	123-456-7892	789 Main St, Yangon	JavaScript, PHP, Swift	2 years	Associate degree
5	JS-004	Alice Williams	Female	alice@example.com	123-456-7893	321 Main St, Yangon	C, C++, Python	1 year	Bachelor degree
6	JS-005	Mike Brown	Male	mike@example.com	123-456-7894	654 Main St, Yangon	Java, C#, Ruby	4 years	Master degree
7	JS-006	Samantha Davis	Female	samantha@example.com	123-456-7895	246 Main St, Mon	JavaScript, PHP, Swift	3 years	Bachelor degree
8	JS-007	William Thompson	Male	william@example.com	123-456-7896	135 Main St, Mon	C, C++, Python	2 years	Associate degree
9	JS-008	Ashley Johnson	Female	ashley@example.com	123-456-7897	753 Main St, Mon	Java, SQL, Ruby	1 year	Bachelor degree
10	JS-009	David Anderson	Male	david@example.com	123-456-7898	159 Main St, Mon	JavaScript, PHP, Swift	5 years	Master degree

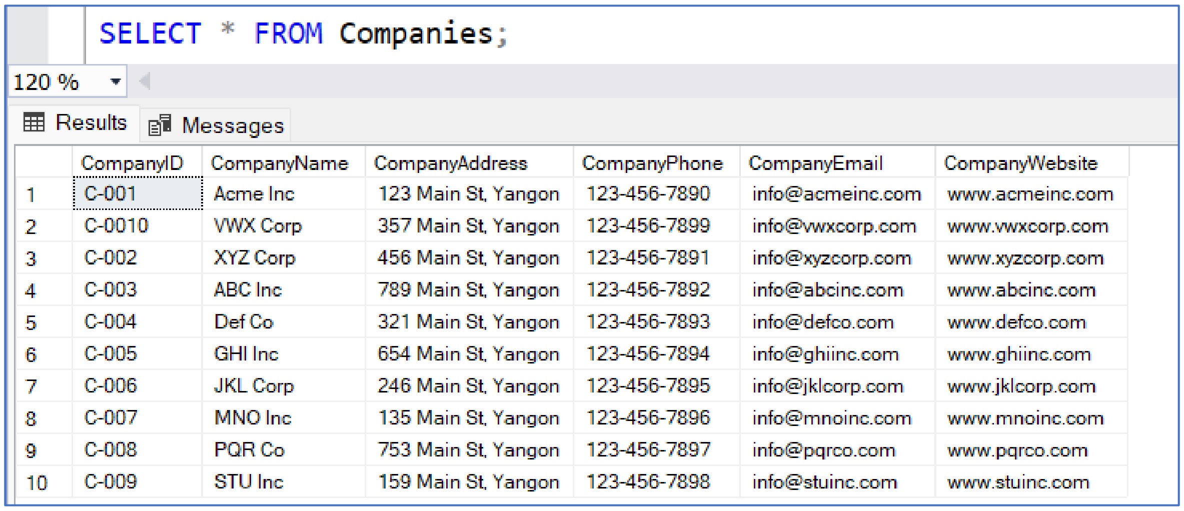
INSERT query and result for Companies



```
INSERT INTO Companies (CompanyID, CompanyName, CompanyAddress, CompanyPhone, CompanyEmail, CompanyWebsite)
VALUES
    ('C-001', 'Acme Inc', '123 Main St, Yangon', '123-456-7890', 'info@acmeinc.com', 'www.acmeinc.com'),
    ('C-002', 'XYZ Corp', '456 Main St, Yangon', '123-456-7891', 'info@xyzcorp.com', 'www.xyzcorp.com'),
    ('C-003', 'ABC Inc', '789 Main St, Yangon', '123-456-7892', 'info@abcinc.com', 'www.abcinc.com'),
    ('C-004', 'Def Co', '321 Main St, Yangon', '123-456-7893', 'info@defco.com', 'www.defco.com'),
    ('C-005', 'GHI Inc', '654 Main St, Yangon', '123-456-7894', 'info@ghiinc.com', 'www.ghiinc.com'),
    ('C-006', 'JKL Corp', '246 Main St, Yangon', '123-456-7895', 'info@jklcorp.com', 'www.jklcorp.com'),
    ('C-007', 'MNO Inc', '135 Main St, Yangon', '123-456-7896', 'info@mnoinc.com', 'www.mnoinc.com'),
    ('C-008', 'PQR Co', '753 Main St, Yangon', '123-456-7897', 'info@pqrco.com', 'www.pqrco.com'),
    ('C-009', 'STU Inc', '159 Main St, Yangon', '123-456-7898', 'info@stuinc.com', 'www.stuinc.com'),
    ('C-0010', 'VWX Corp', '357 Main St, Yangon', '123-456-7899', 'info@vwxcorp.com', 'www.vwxcorp.com');
```

```
(10 rows affected)

Completion time: 2023-01-02T15:22:20.6398657+06:30
```

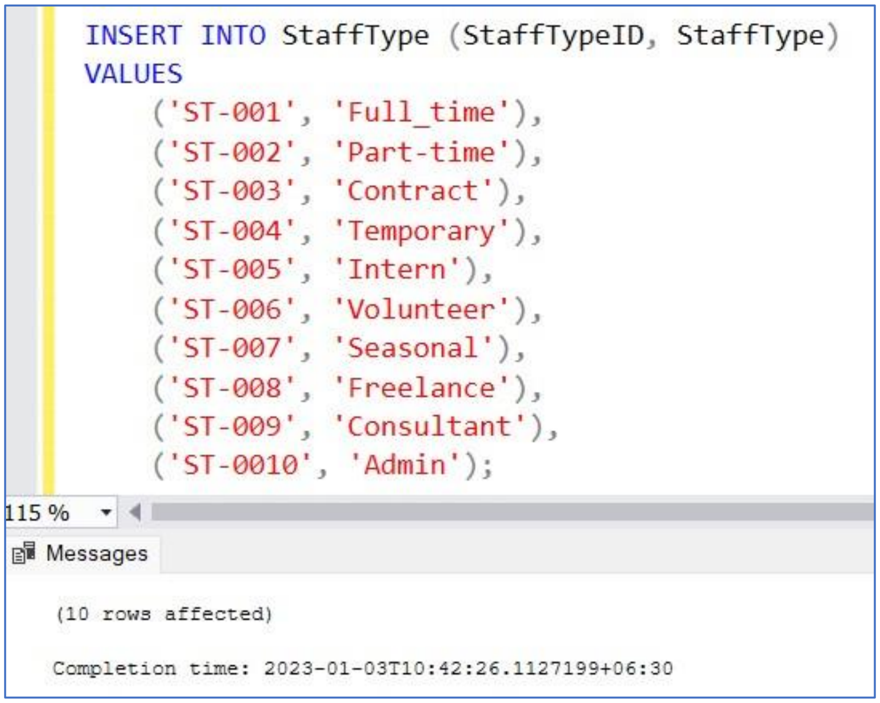
Output for Companies



```
SELECT * FROM Companies;
```

	CompanyID	CompanyName	CompanyAddress	CompanyPhone	CompanyEmail	CompanyWebsite
1	C-001	Acme Inc	123 Main St, Yangon	123-456-7890	info@acmeinc.com	www.acmeinc.com
2	C-0010	VWX Corp	357 Main St, Yangon	123-456-7899	info@vwxcorp.com	www.vwxcorp.com
3	C-002	XYZ Corp	456 Main St, Yangon	123-456-7891	info@xyzcorp.com	www.xyzcorp.com
4	C-003	ABC Inc	789 Main St, Yangon	123-456-7892	info@abcinc.com	www.abcinc.com
5	C-004	Def Co	321 Main St, Yangon	123-456-7893	info@defco.com	www.defco.com
6	C-005	GHI Inc	654 Main St, Yangon	123-456-7894	info@ghiinc.com	www.ghiinc.com
7	C-006	JKL Corp	246 Main St, Yangon	123-456-7895	info@jklcorp.com	www.jklcorp.com
8	C-007	MNO Inc	135 Main St, Yangon	123-456-7896	info@mnoinc.com	www.mnoinc.com
9	C-008	PQR Co	753 Main St, Yangon	123-456-7897	info@pqrco.com	www.pqrco.com
10	C-009	STU Inc	159 Main St, Yangon	123-456-7898	info@stuinc.com	www.stuinc.com

INSERT query and result for StaffType



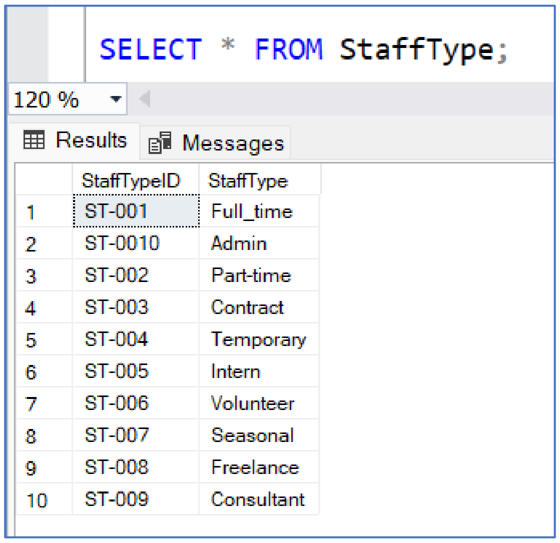
```
INSERT INTO StaffType (StaffTypeID, StaffType)
VALUES
    ('ST-001', 'Full_time'),
    ('ST-002', 'Part-time'),
    ('ST-003', 'Contract'),
    ('ST-004', 'Temporary'),
    ('ST-005', 'Intern'),
    ('ST-006', 'Volunteer'),
    ('ST-007', 'Seasonal'),
    ('ST-008', 'Freelance'),
    ('ST-009', 'Consultant'),
    ('ST-0010', 'Admin');
```

115 %

Messages

```
(10 rows affected)

Completion time: 2023-01-03T10:42:26.1127199+06:30
```

Output for StaffType



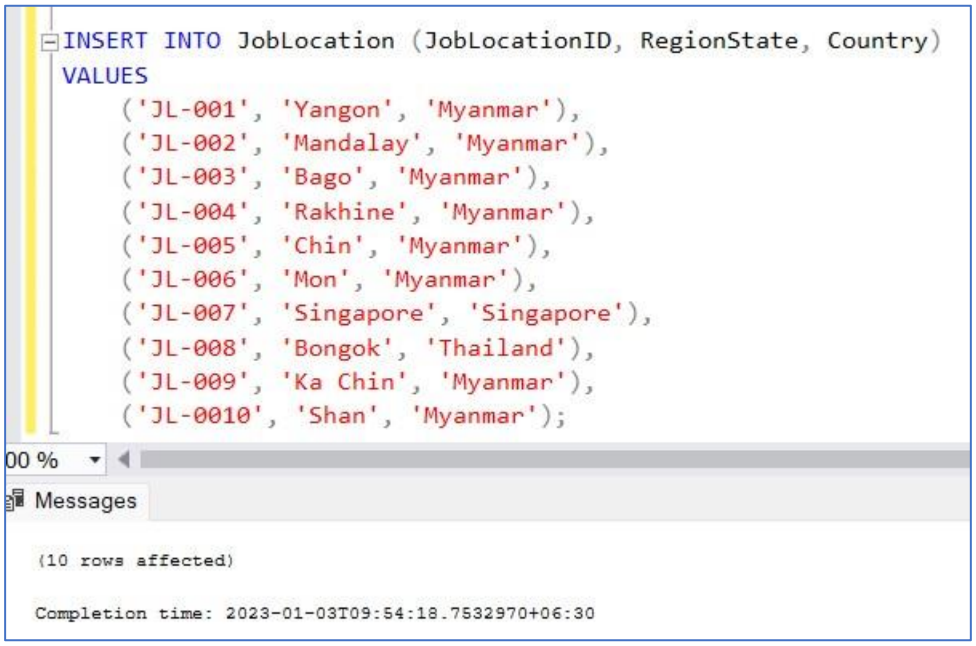
```
SELECT * FROM StaffType;
```

120 %

Results Messages

	StaffTypeID	StaffType
1	ST-001	Full_time
2	ST-0010	Admin
3	ST-002	Part-time
4	ST-003	Contract
5	ST-004	Temporary
6	ST-005	Intern
7	ST-006	Volunteer
8	ST-007	Seasonal
9	ST-008	Freelance
10	ST-009	Consultant

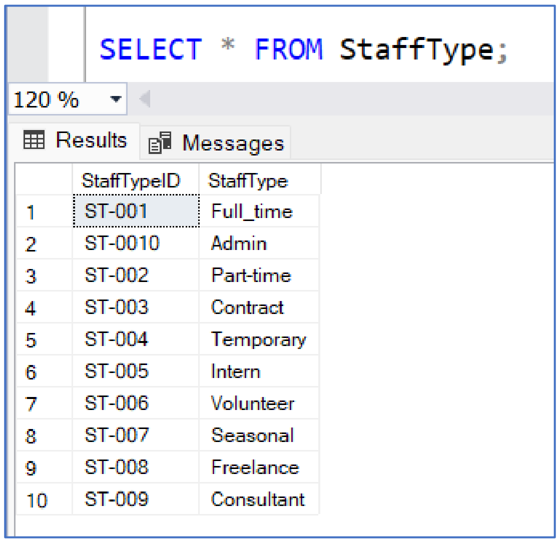
INSERT query and result for JobLocation



```
INSERT INTO JobLocation (JobLocationID, RegionState, Country)
VALUES
    ('JL-001', 'Yangon', 'Myanmar'),
    ('JL-002', 'Mandalay', 'Myanmar'),
    ('JL-003', 'Bago', 'Myanmar'),
    ('JL-004', 'Rakhine', 'Myanmar'),
    ('JL-005', 'Chin', 'Myanmar'),
    ('JL-006', 'Mon', 'Myanmar'),
    ('JL-007', 'Singapore', 'Singapore'),
    ('JL-008', 'Bongok', 'Thailand'),
    ('JL-009', 'Ka Chin', 'Myanmar'),
    ('JL-0010', 'Shan', 'Myanmar');
```

```
(10 rows affected)

Completion time: 2023-01-03T09:54:18.7532970+06:30
```

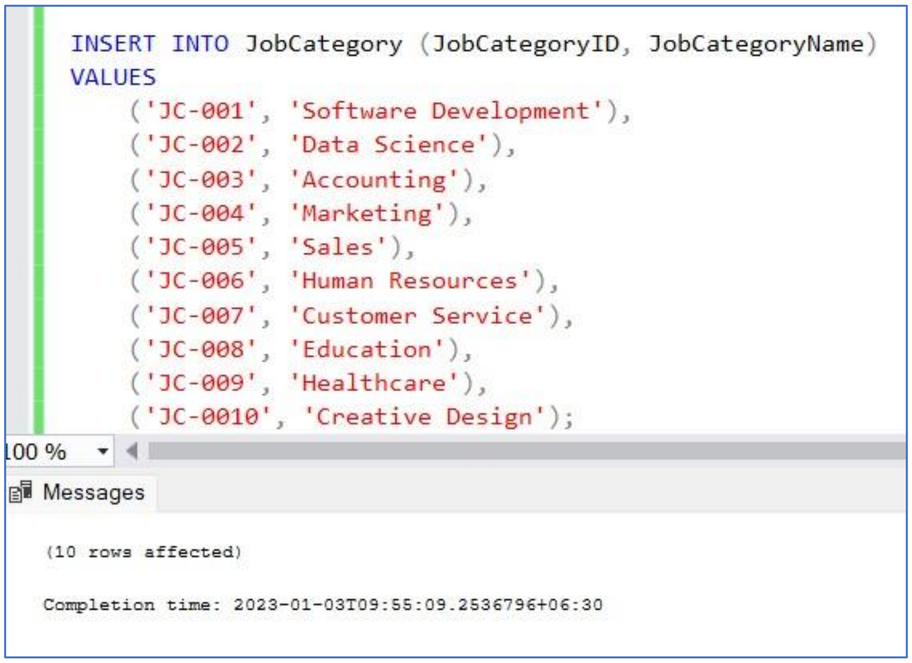
Output for JobLocation



```
SELECT * FROM StaffType;
```

	StaffTypeID	StaffType
1	ST-001	Full_time
2	ST-0010	Admin
3	ST-002	Part-time
4	ST-003	Contract
5	ST-004	Temporary
6	ST-005	Intern
7	ST-006	Volunteer
8	ST-007	Seasonal
9	ST-008	Freelance
10	ST-009	Consultant

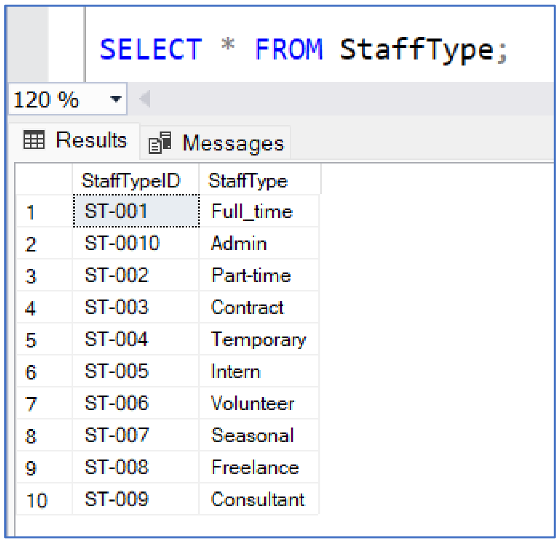
INSERT query and result for JobCategory



```
INSERT INTO JobCategory (JobCategoryID, JobCategoryName)
VALUES
    ('JC-001', 'Software Development'),
    ('JC-002', 'Data Science'),
    ('JC-003', 'Accounting'),
    ('JC-004', 'Marketing'),
    ('JC-005', 'Sales'),
    ('JC-006', 'Human Resources'),
    ('JC-007', 'Customer Service'),
    ('JC-008', 'Education'),
    ('JC-009', 'Healthcare'),
    ('JC-0010', 'Creative Design');
```

```
(10 rows affected)

Completion time: 2023-01-03T09:55:09.2536796+06:30
```

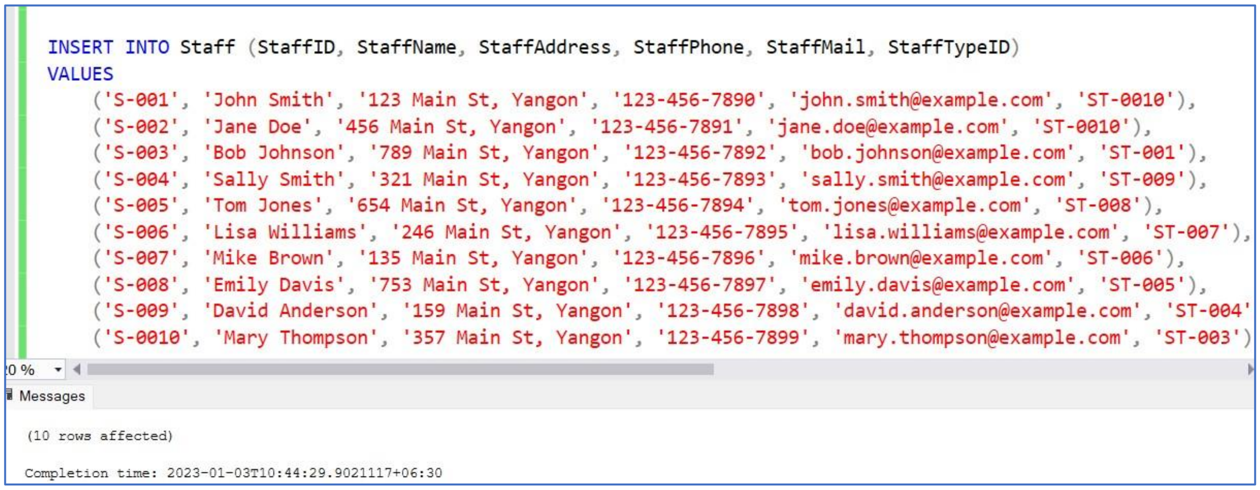
Output for JobCategory



```
SELECT * FROM StaffType;
```

	StaffTypeID	StaffType
1	ST-001	Full_time
2	ST-0010	Admin
3	ST-002	Part-time
4	ST-003	Contract
5	ST-004	Temporary
6	ST-005	Intern
7	ST-006	Volunteer
8	ST-007	Seasonal
9	ST-008	Freelance
10	ST-009	Consultant

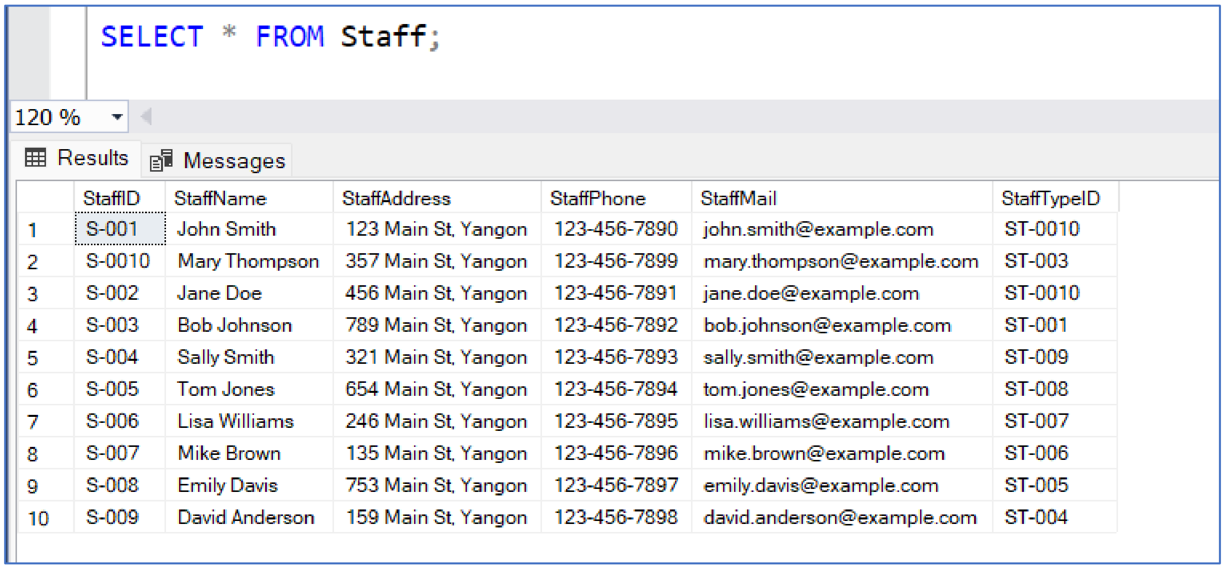
INSERT query and result for Staff



```
INSERT INTO Staff (StaffID, StaffName, StaffAddress, StaffPhone, StaffMail, StaffTypeID)
VALUES
    ('S-001', 'John Smith', '123 Main St, Yangon', '123-456-7890', 'john.smith@example.com', 'ST-0010'),
    ('S-002', 'Jane Doe', '456 Main St, Yangon', '123-456-7891', 'jane.doe@example.com', 'ST-0010'),
    ('S-003', 'Bob Johnson', '789 Main St, Yangon', '123-456-7892', 'bob.johnson@example.com', 'ST-001'),
    ('S-004', 'Sally Smith', '321 Main St, Yangon', '123-456-7893', 'sally.smith@example.com', 'ST-009'),
    ('S-005', 'Tom Jones', '654 Main St, Yangon', '123-456-7894', 'tom.jones@example.com', 'ST-008'),
    ('S-006', 'Lisa Williams', '246 Main St, Yangon', '123-456-7895', 'lisa.williams@example.com', 'ST-007'),
    ('S-007', 'Mike Brown', '135 Main St, Yangon', '123-456-7896', 'mike.brown@example.com', 'ST-006'),
    ('S-008', 'Emily Davis', '753 Main St, Yangon', '123-456-7897', 'emily.davis@example.com', 'ST-005'),
    ('S-009', 'David Anderson', '159 Main St, Yangon', '123-456-7898', 'david.anderson@example.com', 'ST-004'
    ('S-0010', 'Mary Thompson', '357 Main St, Yangon', '123-456-7899', 'mary.thompson@example.com', 'ST-003')
```

```
(10 rows affected)

Completion time: 2023-01-03T10:44:29.9021117+06:30
```

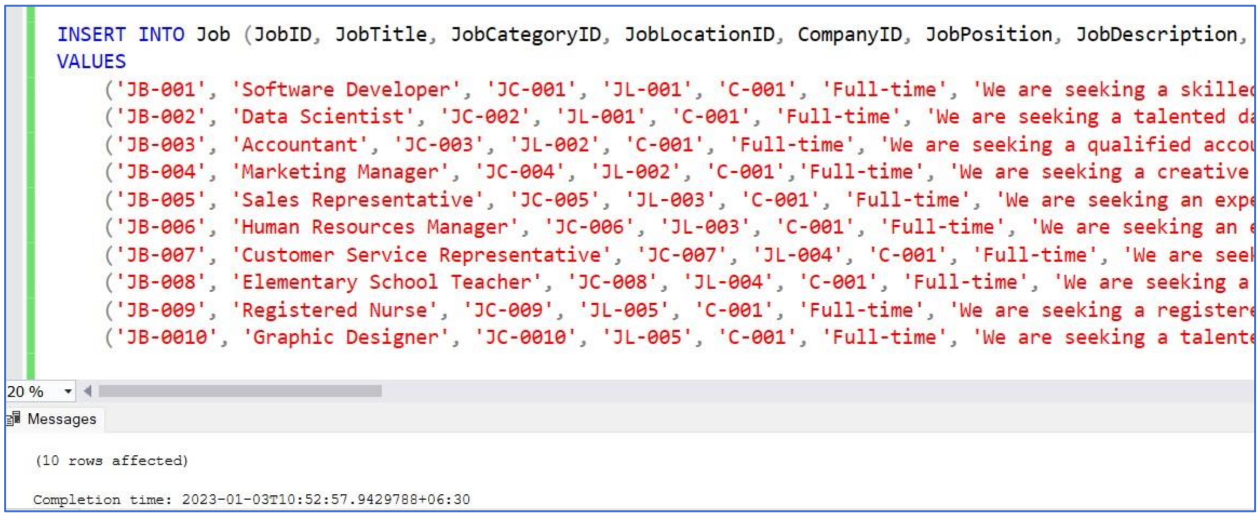
Output for Staff



```
SELECT * FROM Staff;
```

	StaffID	StaffName	StaffAddress	StaffPhone	StaffMail	StaffTypeID
1	S-001	John Smith	123 Main St, Yangon	123-456-7890	john.smith@example.com	ST-0010
2	S-0010	Mary Thompson	357 Main St, Yangon	123-456-7899	mary.thompson@example.com	ST-003
3	S-002	Jane Doe	456 Main St, Yangon	123-456-7891	jane.doe@example.com	ST-0010
4	S-003	Bob Johnson	789 Main St, Yangon	123-456-7892	bob.johnson@example.com	ST-001
5	S-004	Sally Smith	321 Main St, Yangon	123-456-7893	sally.smith@example.com	ST-009
6	S-005	Tom Jones	654 Main St, Yangon	123-456-7894	tom.jones@example.com	ST-008
7	S-006	Lisa Williams	246 Main St, Yangon	123-456-7895	lisa.williams@example.com	ST-007
8	S-007	Mike Brown	135 Main St, Yangon	123-456-7896	mike.brown@example.com	ST-006
9	S-008	Emily Davis	753 Main St, Yangon	123-456-7897	emily.davis@example.com	ST-005
10	S-009	David Anderson	159 Main St, Yangon	123-456-7898	david.anderson@example.com	ST-004

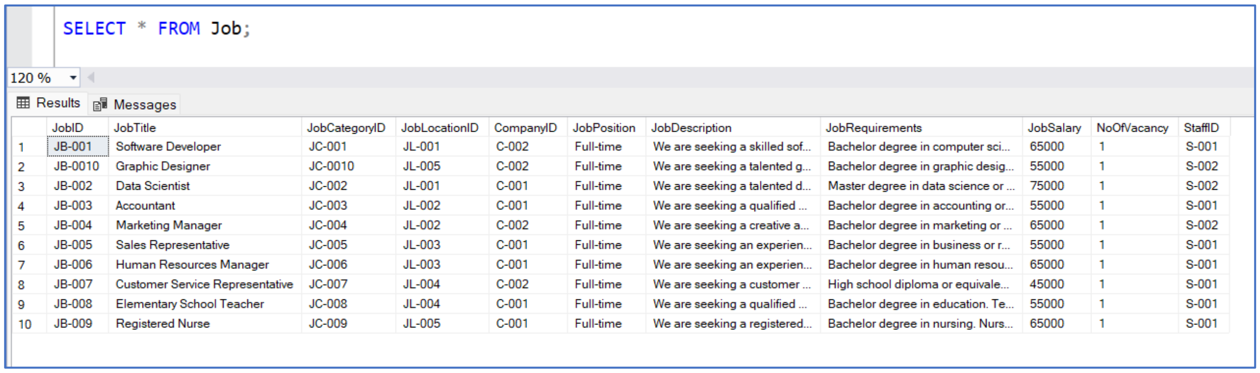
INSERT query and result for Job



```
INSERT INTO Job (JobID, JobTitle, JobCategoryID, JobLocationID, CompanyID, JobPosition, JobDescription,
VALUES
    ('JB-001', 'Software Developer', 'JC-001', 'JL-001', 'C-001', 'Full-time', 'We are seeking a skilled
    ('JB-002', 'Data Scientist', 'JC-002', 'JL-001', 'C-001', 'Full-time', 'We are seeking a talented da
    ('JB-003', 'Accountant', 'JC-003', 'JL-002', 'C-001', 'Full-time', 'We are seeking a qualified accou
    ('JB-004', 'Marketing Manager', 'JC-004', 'JL-002', 'C-001','Full-time', 'We are seeking a creative
    ('JB-005', 'Sales Representative', 'JC-005', 'JL-003', 'C-001', 'Full-time', 'We are seeking an expe
    ('JB-006', 'Human Resources Manager', 'JC-006', 'JL-003', 'C-001', 'Full-time', 'We are seeking an e
    ('JB-007', 'Customer Service Representative', 'JC-007', 'JL-004', 'C-001', 'Full-time', 'We are seek
    ('JB-008', 'Elementary School Teacher', 'JC-008', 'JL-004', 'C-001', 'Full-time', 'We are seeking a
    ('JB-009', 'Registered Nurse', 'JC-009', 'JL-005', 'C-001', 'Full-time', 'We are seeking a registere
    ('JB-0010', 'Graphic Designer', 'JC-0010', 'JL-005', 'C-001', 'Full-time', 'We are seeking a talente
```

```
(10 rows affected)

Completion time: 2023-01-03T10:52:57.9429788+06:30
```

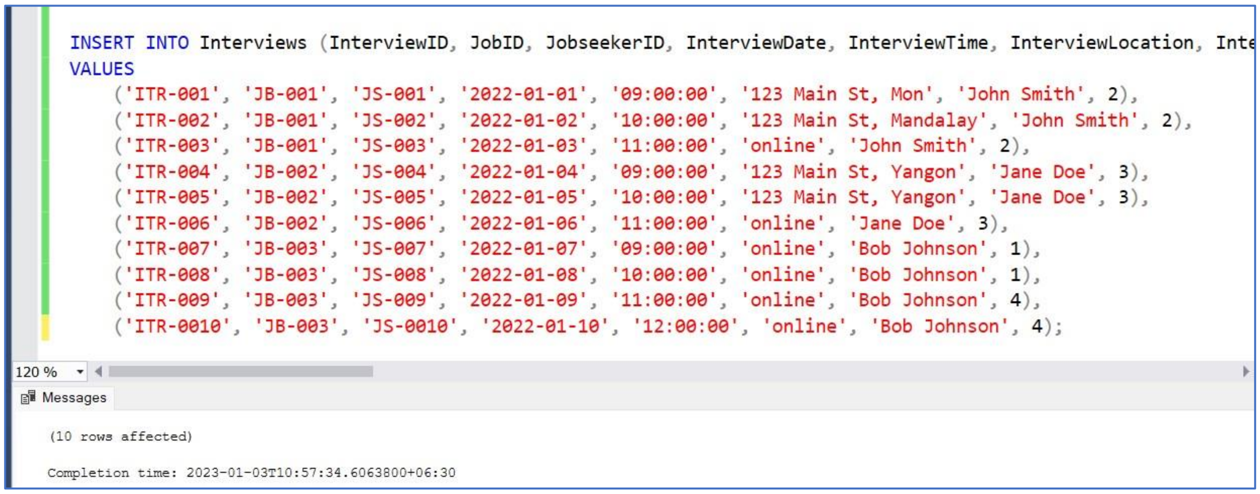
Output for Job



```
SELECT * FROM Job;
```

	JobID	JobTitle	JobCategoryID	JobLocationID	CompanyID	JobPosition	JobDescription	JobRequirements	JobSalary	NoOfVacancy	StaffID
1	JB-001	Software Developer	JC-001	JL-001	C-002	Full-time	We are seeking a skilled sof...	Bachelor degree in computer sci...	65000	1	S-001
2	JB-0010	Graphic Designer	JC-0010	JL-005	C-002	Full-time	We are seeking a talented g...	Bachelor degree in graphic desig...	55000	1	S-002
3	JB-002	Data Scientist	JC-002	JL-001	C-001	Full-time	We are seeking a talented d...	Master degree in data science or ...	75000	1	S-002
4	JB-003	Accountant	JC-003	JL-002	C-001	Full-time	We are seeking a qualified ...	Bachelor degree in accounting or...	55000	1	S-001
5	JB-004	Marketing Manager	JC-004	JL-002	C-002	Full-time	We are seeking a creative a...	Bachelor degree in marketing or ...	65000	1	S-002
6	JB-005	Sales Representative	JC-005	JL-003	C-001	Full-time	We are seeking an experien...	Bachelor degree in business or r...	55000	1	S-001
7	JB-006	Human Resources Manager	JC-006	JL-003	C-001	Full-time	We are seeking an experien...	Bachelor degree in human resou...	65000	1	S-001
8	JB-007	Customer Service Representative	JC-007	JL-004	C-002	Full-time	We are seeking a customer ...	High school diploma or equivale...	45000	1	S-001
9	JB-008	Elementary School Teacher	JC-008	JL-004	C-001	Full-time	We are seeking a qualified ...	Bachelor degree in education. Te...	55000	1	S-001
10	JB-009	Registered Nurse	JC-009	JL-005	C-001	Full-time	We are seeking a registered...	Bachelor degree in nursing. Nurs...	65000	1	S-001

INSERT query and result for Interviews



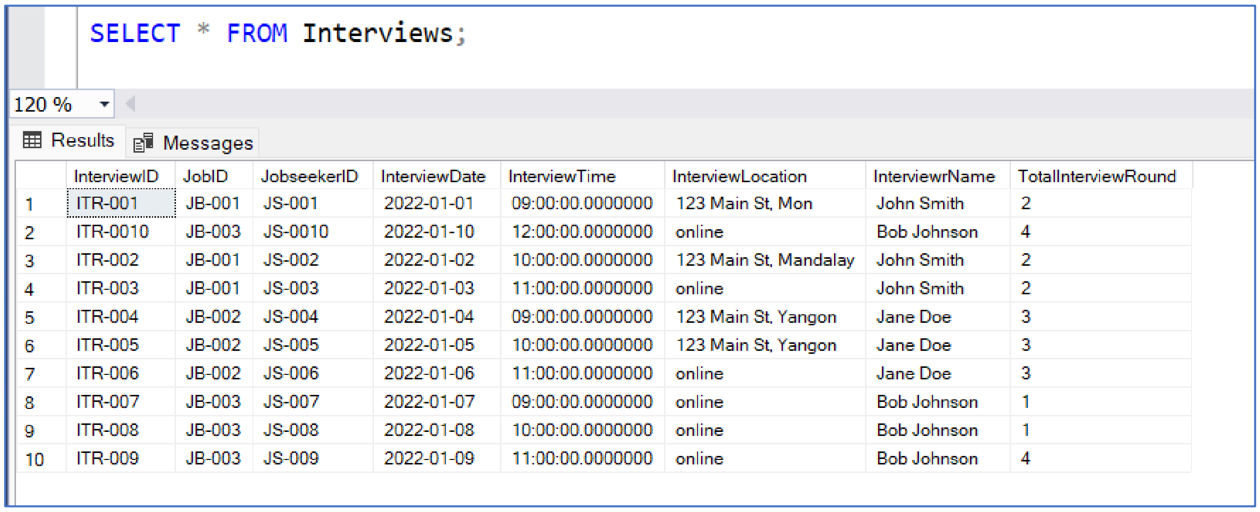
```
INSERT INTO Interviews (InterviewID, JobID, JobseekerID, InterviewDate, InterviewTime, InterviewLocation, Inte
VALUES
    ('ITR-001', 'JB-001', 'JS-001', '2022-01-01', '09:00:00', '123 Main St, Mon', 'John Smith', 2),
    ('ITR-002', 'JB-001', 'JS-002', '2022-01-02', '10:00:00', '123 Main St, Mandalay', 'John Smith', 2),
    ('ITR-003', 'JB-001', 'JS-003', '2022-01-03', '11:00:00', 'online', 'John Smith', 2),
    ('ITR-004', 'JB-002', 'JS-004', '2022-01-04', '09:00:00', '123 Main St, Yangon', 'Jane Doe', 3),
    ('ITR-005', 'JB-002', 'JS-005', '2022-01-05', '10:00:00', '123 Main St, Yangon', 'Jane Doe', 3),
    ('ITR-006', 'JB-002', 'JS-006', '2022-01-06', '11:00:00', 'online', 'Jane Doe', 3),
    ('ITR-007', 'JB-003', 'JS-007', '2022-01-07', '09:00:00', 'online', 'Bob Johnson', 1),
    ('ITR-008', 'JB-003', 'JS-008', '2022-01-08', '10:00:00', 'online', 'Bob Johnson', 1),
    ('ITR-009', 'JB-003', 'JS-009', '2022-01-09', '11:00:00', 'online', 'Bob Johnson', 4),
    ('ITR-0010', 'JB-003', 'JS-0010', '2022-01-10', '12:00:00', 'online', 'Bob Johnson', 4);
```

120 %

Messages

```
(10 rows affected)

Completion time: 2023-01-03T10:57:34.6063800+06:30
```

Output for Interviews



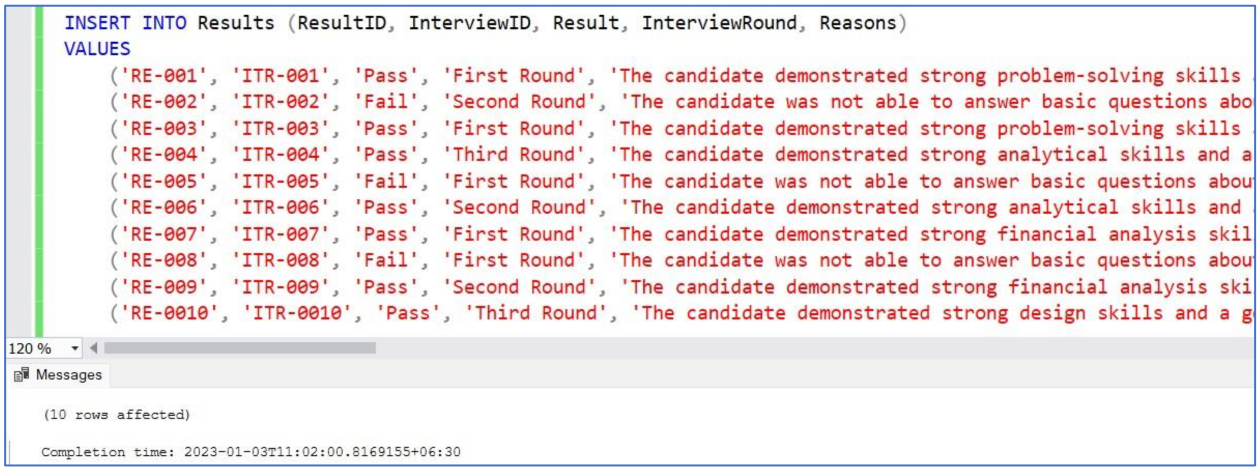
```
SELECT * FROM Interviews;
```

120 %

Results Messages

	InterviewID	JobID	JobseekerID	InterviewDate	InterviewTime	InterviewLocation	InterviewrName	TotalInterviewRound
1	ITR-001	JB-001	JS-001	2022-01-01	09:00:00.0000000	123 Main St, Mon	John Smith	2
2	ITR-0010	JB-003	JS-0010	2022-01-10	12:00:00.0000000	online	Bob Johnson	4
3	ITR-002	JB-001	JS-002	2022-01-02	10:00:00.0000000	123 Main St, Mandalay	John Smith	2
4	ITR-003	JB-001	JS-003	2022-01-03	11:00:00.0000000	online	John Smith	2
5	ITR-004	JB-002	JS-004	2022-01-04	09:00:00.0000000	123 Main St, Yangon	Jane Doe	3
6	ITR-005	JB-002	JS-005	2022-01-05	10:00:00.0000000	123 Main St, Yangon	Jane Doe	3
7	ITR-006	JB-002	JS-006	2022-01-06	11:00:00.0000000	online	Jane Doe	3
8	ITR-007	JB-003	JS-007	2022-01-07	09:00:00.0000000	online	Bob Johnson	1
9	ITR-008	JB-003	JS-008	2022-01-08	10:00:00.0000000	online	Bob Johnson	1
10	ITR-009	JB-003	JS-009	2022-01-09	11:00:00.0000000	online	Bob Johnson	4

INSERT query and result for Results



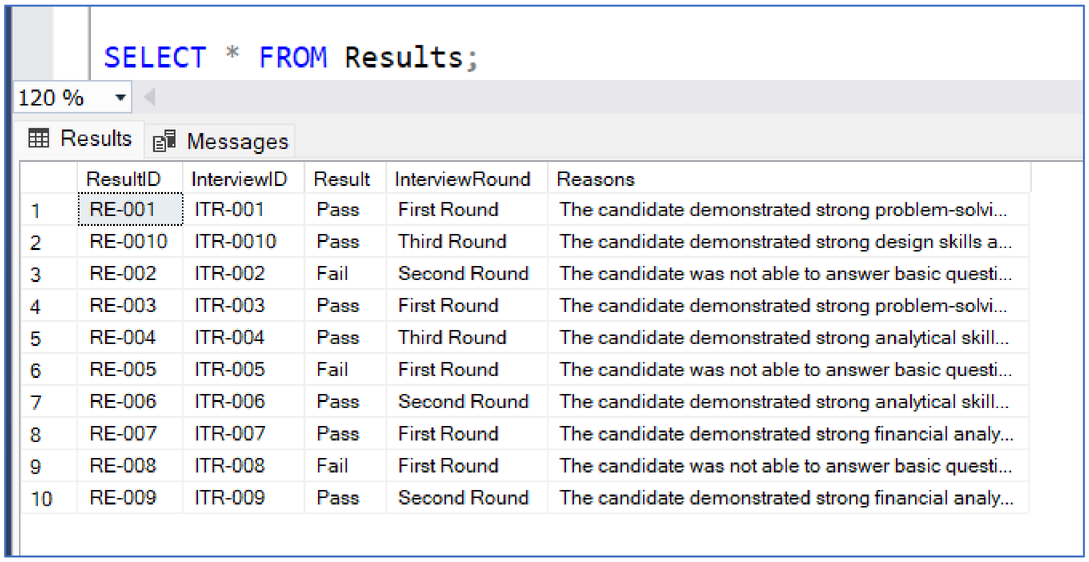
```
INSERT INTO Results (ResultID, InterviewID, Result, InterviewRound, Reasons)
VALUES
    ('RE-001', 'ITR-001', 'Pass', 'First Round', 'The candidate demonstrated strong problem-solving skills 
    ('RE-002', 'ITR-002', 'Fail', 'Second Round', 'The candidate was not able to answer basic questions abo
    ('RE-003', 'ITR-003', 'Pass', 'First Round', 'The candidate demonstrated strong problem-solving skills 
    ('RE-004', 'ITR-004', 'Pass', 'Third Round', 'The candidate demonstrated strong analytical skills and a
    ('RE-005', 'ITR-005', 'Fail', 'First Round', 'The candidate was not able to answer basic questions abou
    ('RE-006', 'ITR-006', 'Pass', 'Second Round', 'The candidate demonstrated strong analytical skills and 
    ('RE-007', 'ITR-007', 'Pass', 'First Round', 'The candidate demonstrated strong financial analysis skil
    ('RE-008', 'ITR-008', 'Fail', 'First Round', 'The candidate was not able to answer basic questions abou
    ('RE-009', 'ITR-009', 'Pass', 'Second Round', 'The candidate demonstrated strong financial analysis ski
    ('RE-0010', 'ITR-0010', 'Pass', 'Third Round', 'The candidate demonstrated strong design skills and a g
```

120 %

Messages

```
(10 rows affected)

Completion time: 2023-01-03T11:02:00.8169155+06:30
```

Output for Results



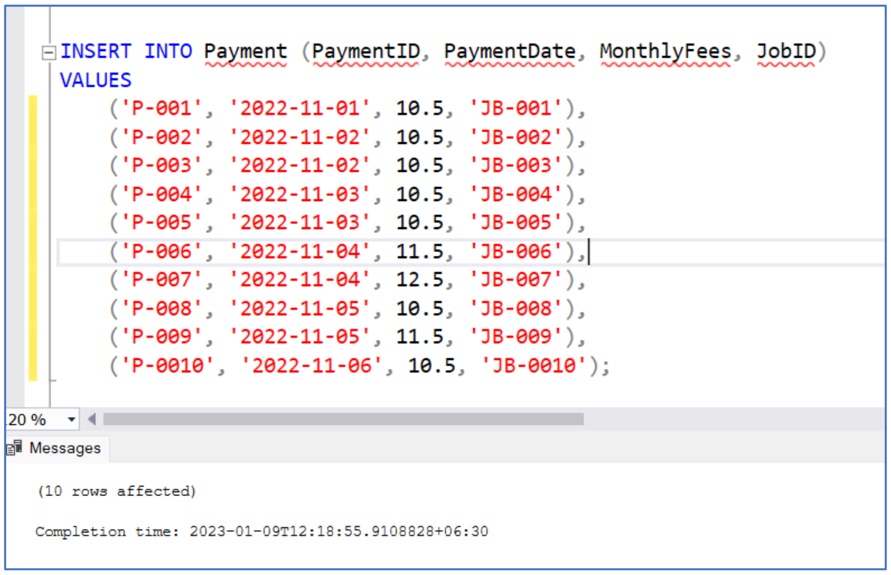
```
SELECT * FROM Results;
```

120 %

Results | Messages

	ResultID	InterviewID	Result	InterviewRound	Reasons
1	RE-001	ITR-001	Pass	First Round	The candidate demonstrated strong problem-solvi...
2	RE-0010	ITR-0010	Pass	Third Round	The candidate demonstrated strong design skills a...
3	RE-002	ITR-002	Fail	Second Round	The candidate was not able to answer basic questi...
4	RE-003	ITR-003	Pass	First Round	The candidate demonstrated strong problem-solvi...
5	RE-004	ITR-004	Pass	Third Round	The candidate demonstrated strong analytical skill...
6	RE-005	ITR-005	Fail	First Round	The candidate was not able to answer basic questi...
7	RE-006	ITR-006	Pass	Second Round	The candidate demonstrated strong analytical skill...
8	RE-007	ITR-007	Pass	First Round	The candidate demonstrated strong financial analy...
9	RE-008	ITR-008	Fail	First Round	The candidate was not able to answer basic questi...
10	RE-009	ITR-009	Pass	Second Round	The candidate demonstrated strong financial analy...

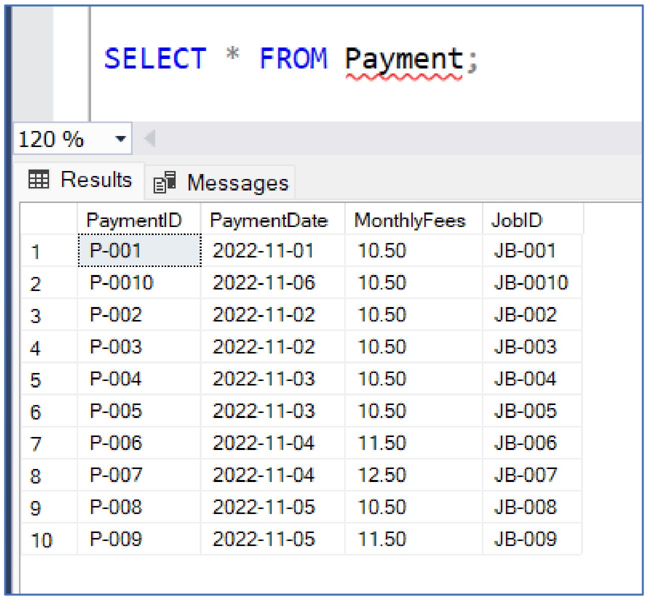
INSERT query and result for Payments



```
INSERT INTO Payment (PaymentID, PaymentDate, MonthlyFees, JobID)
VALUES
    ('P-001', '2022-11-01', 10.5, 'JB-001'),
    ('P-002', '2022-11-02', 10.5, 'JB-002'),
    ('P-003', '2022-11-02', 10.5, 'JB-003'),
    ('P-004', '2022-11-03', 10.5, 'JB-004'),
    ('P-005', '2022-11-03', 10.5, 'JB-005'),
    ('P-006', '2022-11-04', 11.5, 'JB-006'),
    ('P-007', '2022-11-04', 12.5, 'JB-007'),
    ('P-008', '2022-11-05', 10.5, 'JB-008'),
    ('P-009', '2022-11-05', 11.5, 'JB-009'),
    ('P-0010', '2022-11-06', 10.5, 'JB-0010');
```

```
(10 rows affected)

Completion time: 2023-01-09T12:18:55.9108828+06:30
```

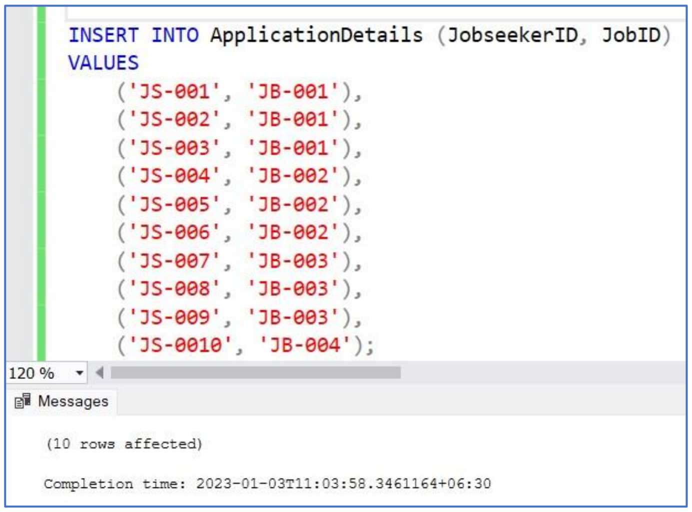
Output for Payment



```
SELECT * FROM Payment;
```

	PaymentID	PaymentDate	MonthlyFees	JobID
1	P-001	2022-11-01	10.50	JB-001
2	P-0010	2022-11-06	10.50	JB-0010
3	P-002	2022-11-02	10.50	JB-002
4	P-003	2022-11-02	10.50	JB-003
5	P-004	2022-11-03	10.50	JB-004
6	P-005	2022-11-03	10.50	JB-005
7	P-006	2022-11-04	11.50	JB-006
8	P-007	2022-11-04	12.50	JB-007
9	P-008	2022-11-05	10.50	JB-008
10	P-009	2022-11-05	11.50	JB-009

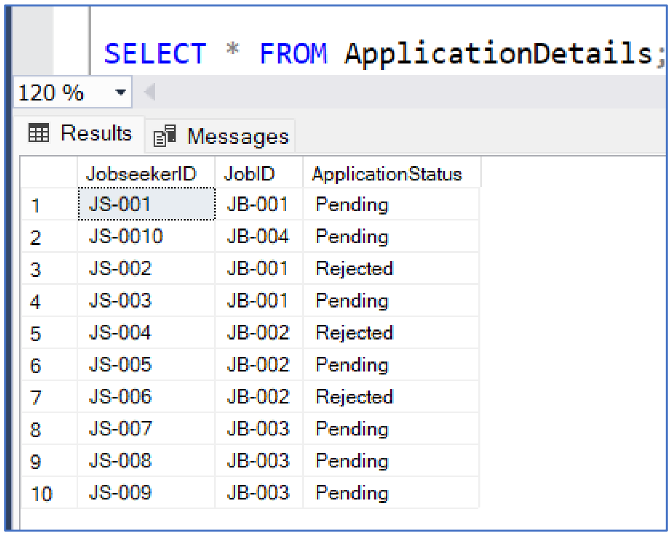
INSERT query and result for ApplicationDetails



```
INSERT INTO ApplicationDetails (JobseekerID, JobID)
VALUES
    ('JS-001', 'JB-001'),
    ('JS-002', 'JB-001'),
    ('JS-003', 'JB-001'),
    ('JS-004', 'JB-002'),
    ('JS-005', 'JB-002'),
    ('JS-006', 'JB-002'),
    ('JS-007', 'JB-003'),
    ('JS-008', 'JB-003'),
    ('JS-009', 'JB-003'),
    ('JS-0010', 'JB-004');
```

```
(10 rows affected)

Completion time: 2023-01-03T11:03:58.3461164+06:30
```

Output for Application Details



```
SELECT * FROM ApplicationDetails;
```

	JobseekerID	JobID	ApplicationStatus
1	JS-001	JB-001	Pending
2	JS-0010	JB-004	Pending
3	JS-002	JB-001	Rejected
4	JS-003	JB-001	Pending
5	JS-004	JB-002	Rejected
6	JS-005	JB-002	Pending
7	JS-006	JB-002	Rejected
8	JS-007	JB-003	Pending
9	JS-008	JB-003	Pending
10	JS-009	JB-003	Pending

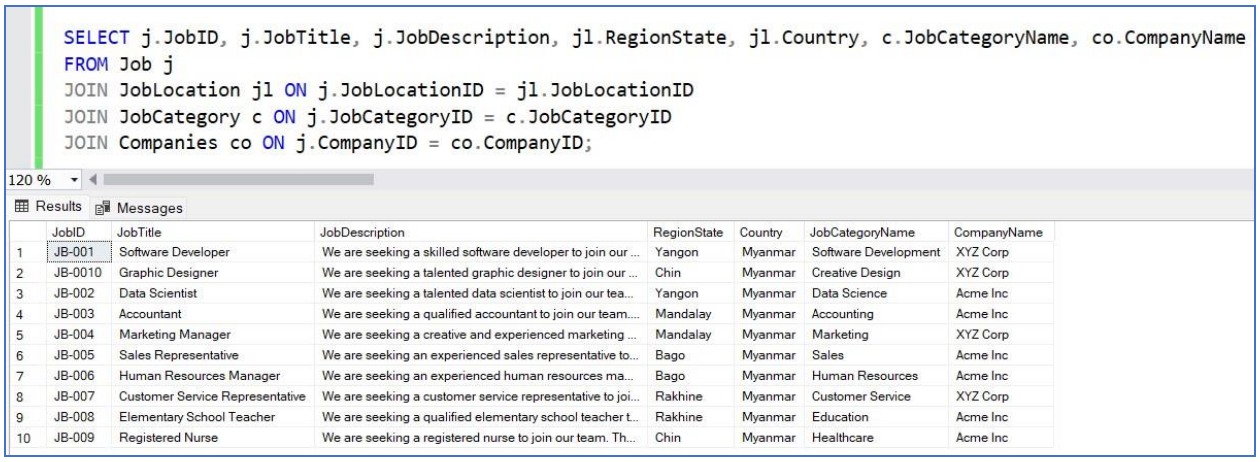
Explanation Summary

To insert data into tables that were created in Task 4, the INERT query, which is contained in the DIL (Data Integrity Language) of SQL, is used. All inert tasks must follow the flow of the table's created arrangement. The data would be entered into each responsive table if we followed that flow. If data is inputted randomly into tables, conflicts may occur with foreign keys as each table is dependent on the tables it belongs to. The difficulty encountered was that the use of foreign keys was too numerous to be used, so many incorrect entries were encountered. Then, when the IDs are ordered, I ID-0010 came first after ID-001, passing over ID-002.

TASK - 6

TASK –6

SQL Reports

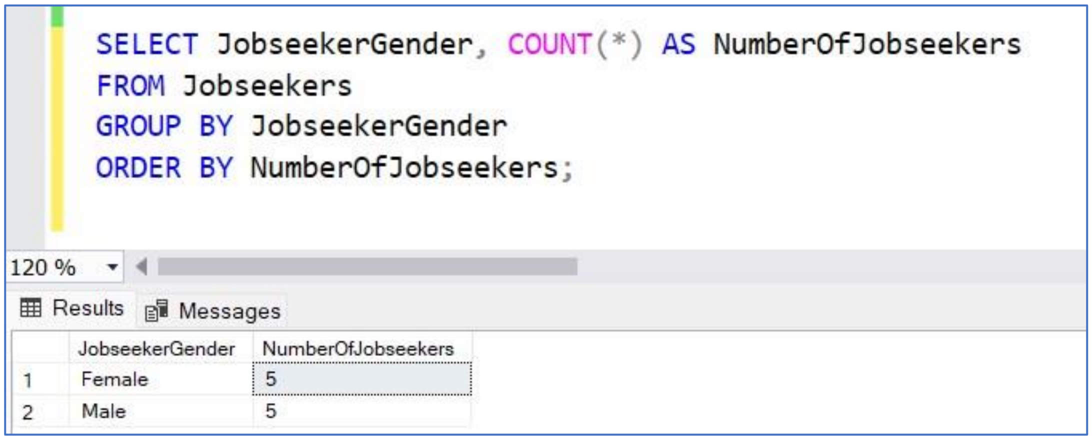


```sql
SELECT j.JobID, j.JobTitle, j.JobDescription, jl.RegionState, jl.Country, c.JobCategoryName, co.CompanyName
FROM Job j
JOIN JobLocation jl ON j.JobLocationID = jl.JobLocationID
JOIN JobCategory c ON j.JobCategoryID = c.JobCategoryID
JOIN Companies co ON j.CompanyID = co.CompanyID;
```

	JobID	JobTitle	JobDescription	RegionState	Country	JobCategoryName	CompanyName
1	JB-001	Software Developer	We are seeking a skilled software developer to join our ...	Yangon	Myanmar	Software Development	XYZ Corp
2	JB-0010	Graphic Designer	We are seeking a talented graphic designer to join our ...	Chin	Myanmar	Creative Design	XYZ Corp
3	JB-002	Data Scientist	We are seeking a talented data scientist to join our tea...	Yangon	Myanmar	Data Science	Acme Inc
4	JB-003	Accountant	We are seeking a qualified accountant to join our team....	Mandalay	Myanmar	Accounting	Acme Inc
5	JB-004	Marketing Manager	We are seeking a creative and experienced marketing ...	Mandalay	Myanmar	Marketing	XYZ Corp
6	JB-005	Sales Representative	We are seeking an experienced sales representative to...	Bago	Myanmar	Sales	Acme Inc
7	JB-006	Human Resources Manager	We are seeking an experienced human resources ma...	Bago	Myanmar	Human Resources	Acme Inc
8	JB-007	Customer Service Representative	We are seeking a customer service representative to joi...	Rakhine	Myanmar	Customer Service	XYZ Corp
9	JB-008	Elementary School Teacher	We are seeking a qualified elementary school teacher t...	Rakhine	Myanmar	Education	Acme Inc
10	JB-009	Registered Nurse	We are seeking a registered nurse to join our team. Th...	Chin	Myanmar	Healthcare	Acme Inc

Figure 1

The SELECT query in Figure 1 is intended to generate a job offered post that end users will see.

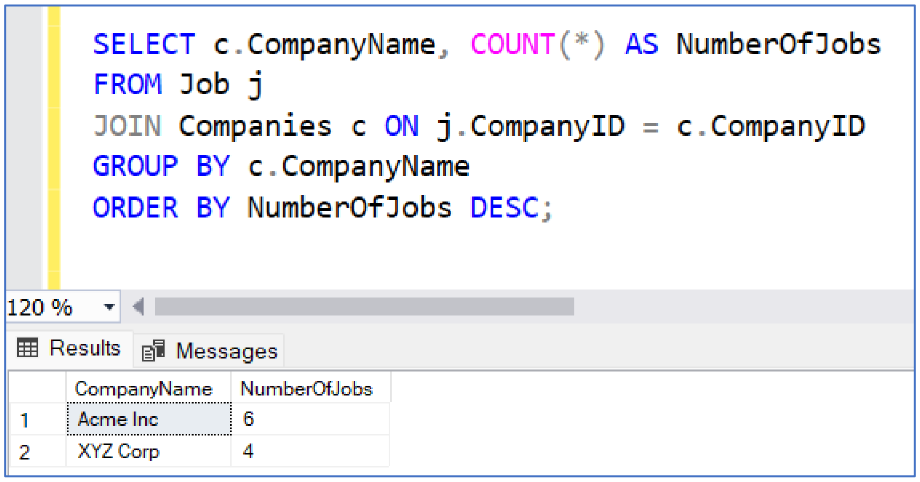


```sql
SELECT JobseekerGender, COUNT(*) AS NumberOfJobseekers
FROM Jobseekers
GROUP BY JobseekerGender
ORDER BY NumberOfJobseekers;
```

	JobseekerGender	NumberOfJobseekers
1	Female	5
2	Male	5

Figure 2

The purpose of figure 2 is to list the female and male job seekers by grouping them.

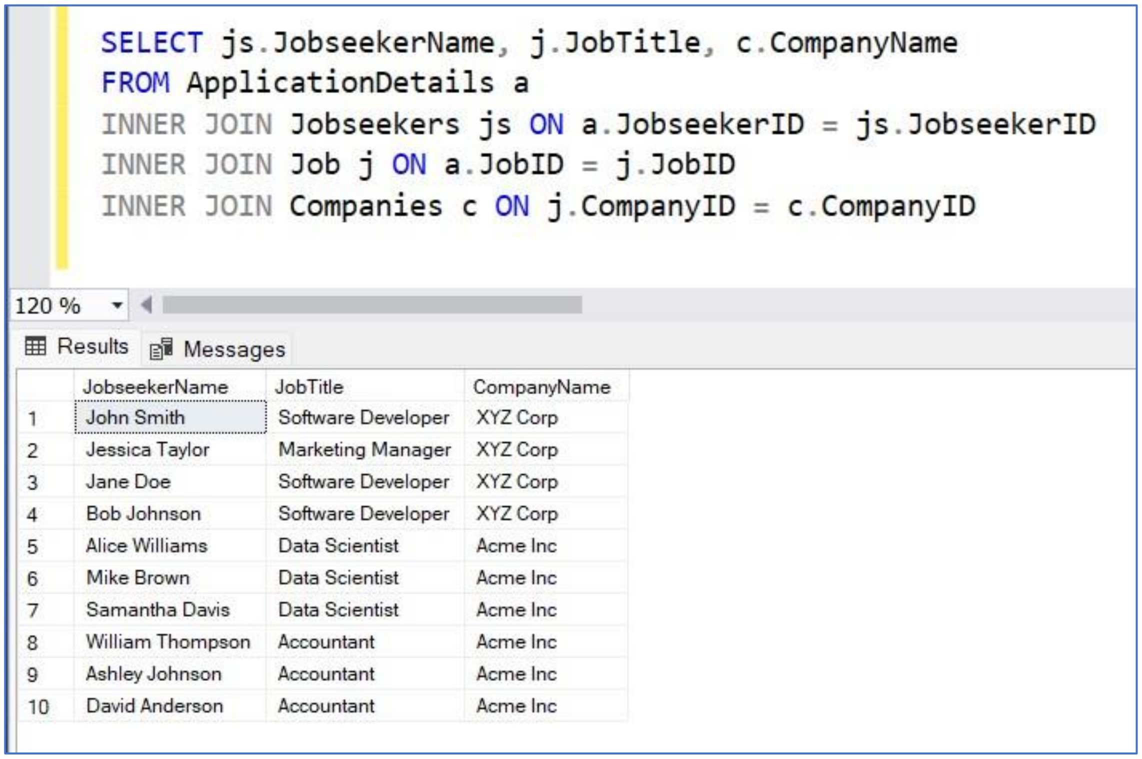


```
SELECT c.CompanyName, COUNT(*) AS NumberOfJobs
FROM Job j
JOIN Companies c ON j.CompanyID = c.CompanyID
GROUP BY c.CompanyName
ORDER BY NumberOfJobs DESC;
```

	CompanyName	NumberOfJobs
1	Acme Inc	6
2	XYZ Corp	4

Figure 3

Looking at the number of job offered listings of each company.

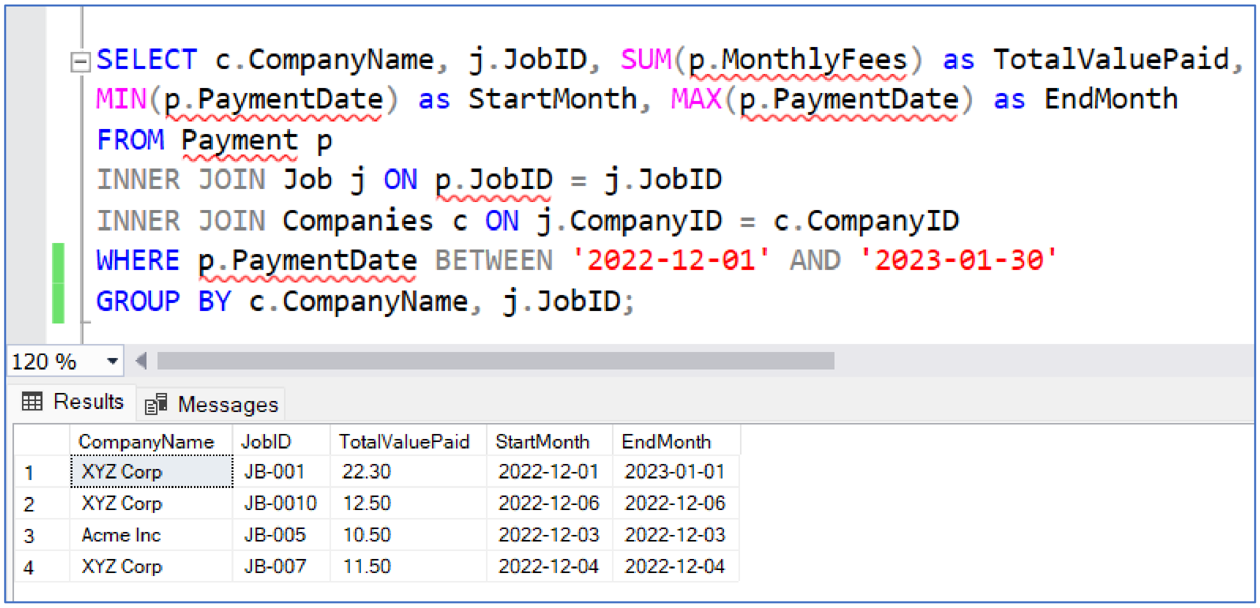


```
SELECT js.JobseekerName, j.JobTitle, c.CompanyName
FROM ApplicationDetails a
INNER JOIN Jobseekers js ON a.JobseekerID = js.JobseekerID
INNER JOIN Job j ON a.JobID = j.JobID
INNER JOIN Companies c ON j.CompanyID = c.CompanyID
```

	JobseekerName	JobTitle	CompanyName
1	John Smith	Software Developer	XYZ Corp
2	Jessica Taylor	Marketing Manager	XYZ Corp
3	Jane Doe	Software Developer	XYZ Corp
4	Bob Johnson	Software Developer	XYZ Corp
5	Alice Williams	Data Scientist	Acme Inc
6	Mike Brown	Data Scientist	Acme Inc
7	Samantha Davis	Data Scientist	Acme Inc
8	William Thompson	Accountant	Acme Inc
9	Ashley Johnson	Accountant	Acme Inc
10	David Anderson	Accountant	Acme Inc

Figure 4

Figure 4 is showing the job application list of each jobseeker.

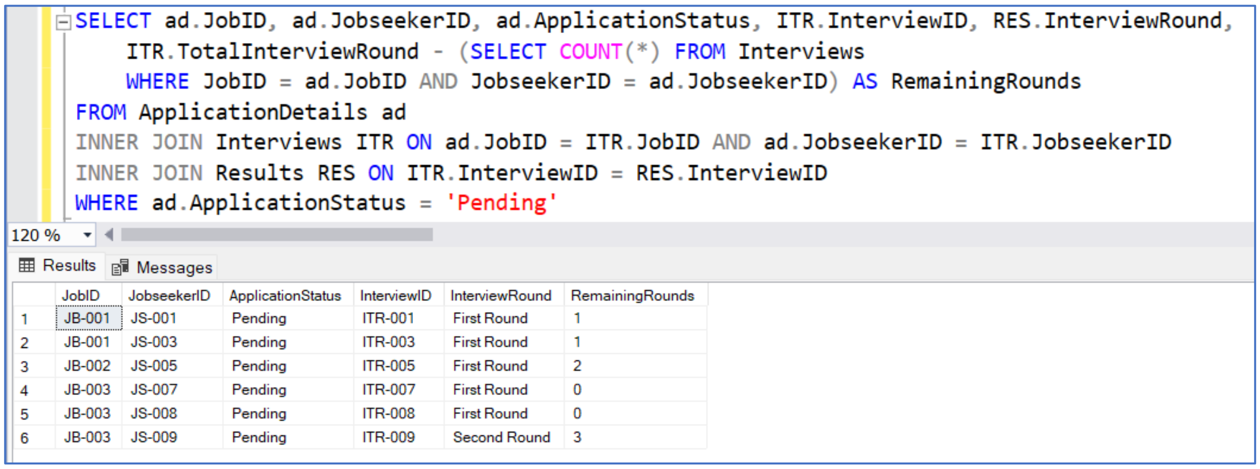


```sql
SELECT c.CompanyName, j.JobID, SUM(p.MonthlyFees) as TotalValuePaid,
MIN(p.PaymentDate) as StartMonth, MAX(p.PaymentDate) as EndMonth
FROM Payment p
INNER JOIN Job j ON p.JobID = j.JobID
INNER JOIN Companies c ON j.CompanyID = c.CompanyID
WHERE p.PaymentDate BETWEEN '2022-12-01' AND '2023-01-30'
GROUP BY c.CompanyName, j.JobID;
```

	CompanyName	JobID	TotalValuePaid	StartMonth	EndMonth
1	XYZ Corp	JB-001	22.30	2022-12-01	2023-01-01
2	XYZ Corp	JB-0010	12.50	2022-12-06	2022-12-06
3	Acme Inc	JB-005	10.50	2022-12-03	2022-12-03
4	XYZ Corp	JB-007	11.50	2022-12-04	2022-12-04

Figure 5

The purpose of figure 5 is to show the total value for each job post paid between December 2022 and January 2023.

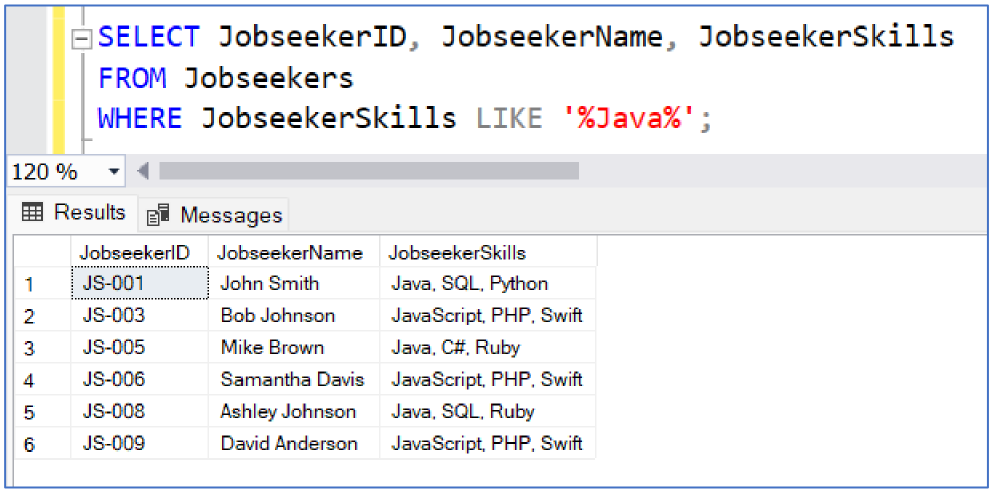


```sql
SELECT ad.JobID, ad.JobseekerID, ad.ApplicationStatus, ITR.InterviewID, RES.InterviewRound,
    ITR.TotalInterviewRound - (SELECT COUNT(*) FROM Interviews
    WHERE JobID = ad.JobID AND JobseekerID = ad.JobseekerID) AS RemainingRounds
FROM ApplicationDetails ad
INNER JOIN Interviews ITR ON ad.JobID = ITR.JobID AND ad.JobseekerID = ITR.JobseekerID
INNER JOIN Results RES ON ITR.InterviewID = RES.InterviewID
WHERE ad.ApplicationStatus = 'Pending'
```

	JobID	JobseekerID	ApplicationStatus	InterviewID	InterviewRound	RemainingRounds
1	JB-001	JS-001	Pending	ITR-001	First Round	1
2	JB-001	JS-003	Pending	ITR-003	First Round	1
3	JB-002	JS-005	Pending	ITR-005	First Round	2
4	JB-003	JS-007	Pending	ITR-007	First Round	0
5	JB-003	JS-008	Pending	ITR-008	First Round	0
6	JB-003	JS-009	Pending	ITR-009	Second Round	3

Figure 6

The purpose of figure 6 is to generate a list of pending applications status and remaining rounds for interview.

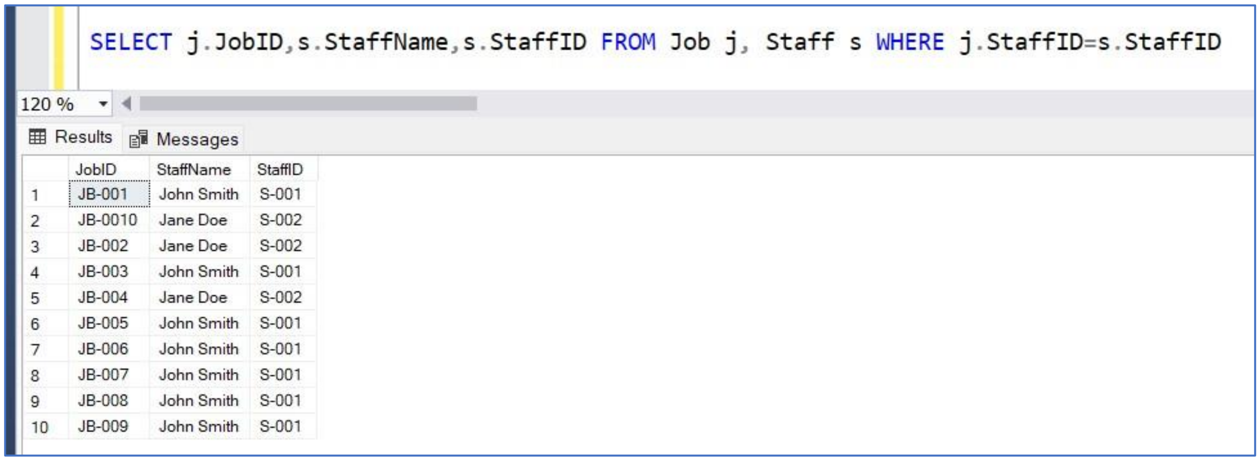


```
SELECT JobseekerID, JobseekerName, JobseekerSkills
FROM Jobseekers
WHERE JobseekerSkills LIKE '%Java%';
```

	JobseekerID	JobseekerName	JobseekerSkills
1	JS-001	John Smith	Java, SQL, Python
2	JS-003	Bob Johnson	JavaScript, PHP, Swift
3	JS-005	Mike Brown	Java, C#, Ruby
4	JS-006	Samantha Davis	JavaScript, PHP, Swift
5	JS-008	Ashley Johnson	Java, SQL, Ruby
6	JS-009	David Anderson	JavaScript, PHP, Swift

Figure 7

Figure 7 is a list of people who are proficient in java and JavaScript.

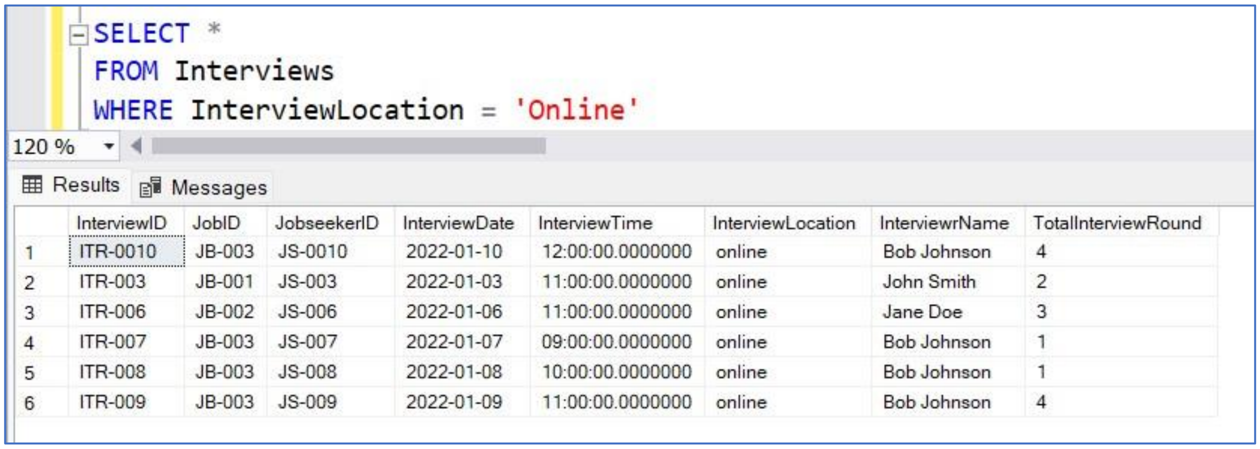


```
SELECT j.JobID,s.StaffName,s.StaffID FROM Job j, Staff s WHERE j.StaffID=s.StaffID
```

	JobID	StaffName	StaffID
1	JB-001	John Smith	S-001
2	JB-0010	Jane Doe	S-002
3	JB-002	Jane Doe	S-002
4	JB-003	John Smith	S-001
5	JB-004	Jane Doe	S-002
6	JB-005	John Smith	S-001
7	JB-006	John Smith	S-001
8	JB-007	John Smith	S-001
9	JB-008	John Smith	S-001
10	JB-009	John Smith	S-001

Figure 8

Figure 8 is a list of staff who approved each job.



```
SELECT *
FROM Interviews
WHERE InterviewLocation = 'Online'
```

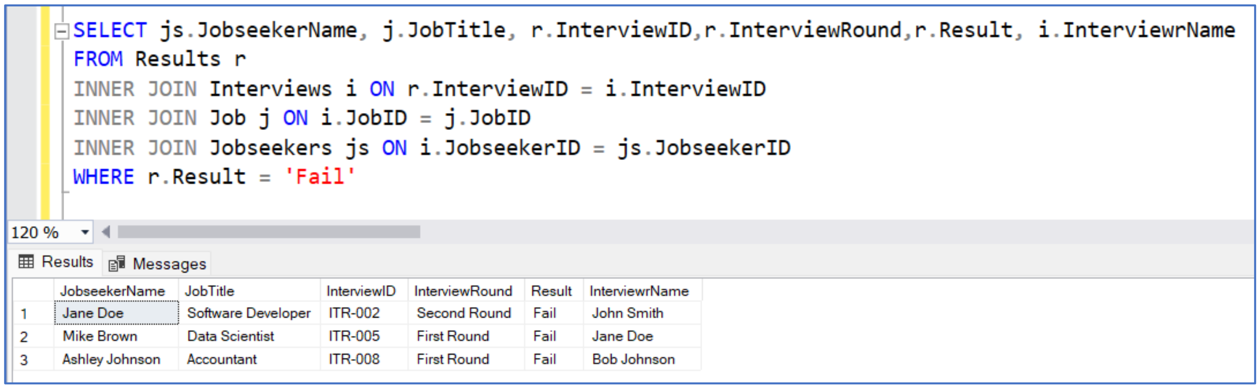
120 %

Results | Messages

	InterviewID	JobID	JobseekerID	InterviewDate	InterviewTime	InterviewLocation	InterviewrName	TotalInterviewRound
1	ITR-0010	JB-003	JS-0010	2022-01-10	12:00:00.0000000	online	Bob Johnson	4
2	ITR-003	JB-001	JS-003	2022-01-03	11:00:00.0000000	online	John Smith	2
3	ITR-006	JB-002	JS-006	2022-01-06	11:00:00.0000000	online	Jane Doe	3
4	ITR-007	JB-003	JS-007	2022-01-07	09:00:00.0000000	online	Bob Johnson	1
5	ITR-008	JB-003	JS-008	2022-01-08	10:00:00.0000000	online	Bob Johnson	1
6	ITR-009	JB-003	JS-009	2022-01-09	11:00:00.0000000	online	Bob Johnson	4

Figure 9

Figure 9 is for online interview list.



```
SELECT js.JobseekerName, j.JobTitle, r.InterviewID,r.InterviewRound,r.Result, i.InterviewrName
FROM Results r
INNER JOIN Interviews i ON r.InterviewID = i.InterviewID
INNER JOIN Job j ON i.JobID = j.JobID
INNER JOIN Jobseekers js ON i.JobseekerID = js.JobseekerID
WHERE r.Result = 'Fail'
```

120 %

Results | Messages

	JobseekerName	JobTitle	InterviewID	InterviewRound	Result	InterviewrName
1	Jane Doe	Software Developer	ITR-002	Second Round	Fail	John Smith
2	Mike Brown	Data Scientist	ITR-005	First Round	Fail	Jane Doe
3	Ashley Johnson	Accountant	ITR-008	First Round	Fail	Bob Johnson

Figure 10

Figure 10 is listing job seekers who fail interview.

TASK - 7

TASK – 7

1.1 Mapped logical database design to physical database design

There were three types when mapping logical to physical design. They are many to many, one to one and one to many. After normalizing and checking the resulting table structures, tables can be created as a physical design.

1.1.1 Many to many (Entities to table – 1)

The entities named Jobseekers and Jobs are connecting by many to many. There was a reason a job seeker can apply many jobs and a job can be applied by many job seekers. That is why a dummy table was created between them.

1.1.2 One to one (Entities to table -2)

The entities named Jobseekers and resumes are connecting by one to one. One to one table is connecting primary key each other. So, it didn't need to sperate them. That is why resumes entity was added to Jobseekers entity.

1.1.3 One to many (Entities to table-3)

One-to-many tables are where one entity has a unique relationship with another and affects many of them independently. If only one data in a one domain table changes, the data in the domain table that was supported by one domain will also change dynamically. The tables used for one to many are JobCategory to Jobs, JobLocations to Jobs, Job seekers to Applications Details, StaffTypes to Staff, Staff to Jobs, Companies to Jobs, Jobs to Payments, Jobs to Interviews, Jobseekers to interviews and Interviews to Results.

1.2 Designed tables for your target DBMS

Microsoft SQL Server Management's version 18.12.1 was used as a DBSM. The database scripting language was SQL. To create and delete tables, CREATE and DROP queries were used, which were from DDL (Data Definition Language). After creating the tables, I used the check key word that ensures the domain constraints (for example, JobID = JB-001) were unique. I also used propagation constraints on tables with foreign keys. To change data types and delete the rows and columns, ALTER (DDL) and DELETE (DIL, Data Manipulation Language) queries were used suitably. To control data, INSERT, UPDATE, DELETE, and SELECT queries that are DCL (Data Control Language) were used. When using Alter queries, conditional cases such as IF and ELSE are used. To calculate data, functions such as MAX(), MIN(), COUNT(), etc. were used. WHERE queries were used in conjunction with restricting data from tables, SELECT queries, and to check condition WHERE queries.

1.3 Derived Data

Derived data is a value that is obtained from a source.

1.3.1 Derived Data (1)

```
ALTER TABLE Payments

ADD DailyAvgFees decimal(10,2)


UPDATE Payments

SET DailyAvgFees = MonthlyFees / 30

Where Payments JobID = Job JobID
```

Deriving is used in Payments table. In early stage, there is no column for DailyAvgFees. So a new column named as a DailyAvgFees is added. The main purpose is to compute the valuate that is calculating each monthly fees into daily average fees by dividing with 30. That query will be work when the condition is when JobID from Payments and Job table are same.

1.3.2 Derived Data (2)

```
ALTER TABLE Result

ADD CompletedRounds INT;


ALTER TABLE Result

ADD RemainingRounds INT;


UPDATE Result

SET RemainingRounds = TotalRounds - CompletedRounds

WHERE Interviews InterviewID = Results InterviewID;
```

Further derivation is used in the Results table. The usage is to add two columns named CompletedRounds and RemainingRounds in the first case. After that, we have to insert the existing data condition, so we can use UPDATE query and subtract CompletedRounds from TotalRounds to get the deriving data.

1.4 Describing about the set of queries that have utility for the business

Restricted queries useful for business are mentioned in task-6.

The query written for figure (1) is written for a job offered post.

It is written in figure (2) so that the number of people who come to work can be divided into male and female.

It was in figure (3) to be able to see the list of total jobs of a company.

It was written n figure (4) to see the job application list of each jobseeker.

For each job offered on the moon job online platform, the total values with the name of the company are shown in figure (5) so that end user can see the total value for a limited time.

It can be seen in figure (6) that those who have passed the previous rounds in the interview are written so that they can know the remaining rounds.

It was written in figure (7) to find people who are proficient in Java and JavaScript.

To know the name of each staff approved for each job, it was written in figure (8).

The list of people who will be interviewed online is shown in figure (9).

The last figure (10) was written to describe the interview fail list.

1.5 Writing a report on whether the points outlined in task (1) are met

As described in Task 1, the system was able to build a database for the Moon Job Online Search Platform. It was very useful for normalizing forms later in Task 2 because the text in Task 1 had to be written in transliterations. The sample documentation is included in Task 1, and we had to look at those documents and normalize them. When all of the entity relationship diagrams that resulted from the normalization were combined, a complete system database was produced. In task (1), some texts had to be removed because they were not needed. For example, tables with a 1 to1 relationship are not needed, so when they were made into a single table, they did not match what was described in Task 1, so it was necessary to adjust some of the text in the scenario. However, the final physical design results are consistent with those described in Task 1.

TASK - 8

TASK – 8

Future Development of a data ware house

A data warehouse is a repository for structured data used for reporting and analysis, optimized for fast querying and analysis. It is often used to support decision-making in organizations and is useful for consolidating data from multiple sources. Moon, a job portal, may build a data warehouse in the future to centralize data from multiple sources and easily analyze and gain insights. Data warehouses are typically used by organizations with large amounts of data, a need for advanced analytics, or a requirement for real-time or near-real-time analysis. Examples include retailers, healthcare providers, and financial institutions. There are two types of database processing: OLTP (Online Transaction Processing) systems support the fast processing of high volumes of transactions, while OLAP (Online Analytical Processing) systems are used for fast querying and analysis of large datasets, often for business intelligence and data analysis.

To build a successful data warehouse, it's important to: Understand the needs of the end user. Identify data sources. analyze the obtained sources. Use data transformation information. Create meta data to describe integration and transformation. Construct a physical data warehouse and populate it with various sources.

There are four steps to input data from the OLTP system to the data warehouse. These can be called key features of a data warehouse. To input data from an OLTP system into a data warehouse, there are four key steps to consider: Integration: converting data from various sources into a consistent format, such as changing the gender format of jobseekers from "male" and "female" in the OLTP system to "m" and "f" in the data warehouse. Time-variance: ensuring that data is stored for a specific period of time, such as keeping track of monthly payments for each job for a 3-year horizon in the OLTP system but discarding this data once the 3 years have passed. Non-volatility: ensuring that data in the warehouse is read-only to prevent it from being accidentally modified or deleted. Subject-orientation: organizing the data warehouse around specific subjects such as job seekers, employers, and job listings rather than around specific applications or data sources This allows for more efficient querying and analysis of the data.

A data warehouse functional model for the Moon job portal will involve extracting data from various sources, such as job board websites, social media platforms, and email servers. The data would then be transformed into a consistent format and loaded into the data warehouse. The data warehouse would be optimized for fast querying and analysis and could be used to support business intelligence and data-driven decision making. Tools such as SQL or a business intelligence platform could be used to query and analyze the data, and the results could be used to inform business decisions such as adjusting algorithms or targeting marketing campaigns.

(Taylor, January 5, 2023) (Anon., n.d.)

TASK - 9

TASK – 9

Distributed Database Option

In order to support the future expansion and growth of our organization, Moon Job Portal, we are considering implementing a distributed database system. The distributed database is that spreading across the multiple servers, locations, or devices and allows the multiple users to access and modify the data simultaneously. Fragmentation and replication might be involved in this.

There are many components of a distributed database management system (DDBMS) in which server process, a distributed database, a network and client applications. Various factors will be need to consider, in order to effectively apply a distributed database, such as current organizational, the use of replication, fragmentation, and different types of distributed database.

In order to maintain the transparency level of the system, it is planned to use a transparent distributed database. It collects all the data in one place and stands as a single. Another factor is that if there is not much risk between different locations and servers, to improve the communication and coordination level, we will only use a homogeneous distributed database. That type of database must use the same database management system (DBMS) in different places and different sites.

If it is going to implement a distributed database, it should first collect and organize the specified data. And then it has to insert into the table using the INSERT INTO statements that will populate the database and data. The code should be run and tested in a database environment.

Implementing a distributed database system has the potential to greatly enhance the capabilities of our organization, Moon Job Portal. A distributed database can improve performance and scalability by allowing data to be stored and accessed on multiple servers or devices simultaneously, and can also improve data availability by allowing our organization to continue operating even if one of the servers goes offline. However, it is important to note that implementing a distributed database can also be complex and costly, and may require additional hardware and software resources.

Additionally, it can be more challenging to monitor and maintain a distributed database, as it requires ongoing attention to ensure that all servers and locations are functioning properly. Despite these potential drawbacks, we believe that the benefits of implementing a distributed database outweigh the challenges. By carefully planning and executing the implementation process, we hope to successfully implement a distributed database that will support the growth and expansion of our organization, Moon Job Portal.

(Moore, n.d.)

References

Anon., 2015. *The Pros and Cons of Data Warehouses.* [Online]
Available at: https://businessimpactinc.com/blog/the-pros-cons-of-data-warehouses/
[Accessed 1 August 2022].

Anon., n.d. *Data Warehouse Concepts, Architecture, and Comparisons.* [Online]
Available at: https://streamsets.com/learn/data-warehouse/
[Accessed 7 Jun 2023].

Biscobing, J., n.d. *What is Entity Relationship Diagram (ERD)?.* [Online]
Available at: https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD
[Accessed 29 June 2022].

Brown, M., 2021. *What are the differences between a Data Warehouse and a Transactional Database?.* [Online]
Available at: https://waterloodata.com/differences-between-data-warehouse-and-transactional-database/?fbclid=IwAR0nb5wVYtQPNPZjnk4Cac9-u2tcS7ceC9FbfaKOoDaRFFS43xkjyZS72oQ
[Accessed 1 August 2022].

Derda, M., 2020. *What is a data dictionary?.* [Online]
Available at: https://www.trifacta.com/blog/data-dictionary/#:~:text=Matt%20Derda
[Accessed 28 June 2022].

Dhyawala, S., 2018. *Denormalization in Databases.* [Online]
Available at: https://www.geeksforgeeks.org/denormalization-in-databases
[Accessed 8 July 2022].

Gierc, M., 2019. *Why your business needs a data warehouse.* [Online]
Available at: https://www.datasciencecentral.com/why-your-business-needs-a-data-warehouse/
[Accessed 26 July 2022].

IBM, 2021. *Resolve m:n relationships.* [Online]
Available at: https://www.ibm.com/docs/en/informix-servers/14.10?topic=relationships-resolve-mn
[Accessed 8 July 2022].

Jennifer, n.d. *What Is A Well Structured Relation In Database Design?.* [Online]
Available at: https://www.rkimball.com/what-is-a-well-structured-relation-in-database-design/
[Accessed 10 July 2022].

Moore, L., n.d. *distributed database.* [Online]
Available at: https://www.techtarget.com/searchoracle/definition/distributed-database#:~:text=A%20distributed%20database%20is%20a,distributed%20among%20multiple%20database%20nodes.
[Accessed 31 Dec 2022].

SINGH, C., 2015. *Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database.* [Online]
Available at: https://beginnersbook.com/2015/05/normalization-in-dbms
[Accessed 5 July 2022].

Taylor, D., January 5, 2023. *What is Data Warehouse? Types, Definition & Example.* [Online]
Available at: https://www.guru99.com/data-warehousing.html#9
[Accessed 7 Jun 2023].

Watt, A., 2014. *Chapter 12 Normalization.* [Online]
Available at: https://opentextbc.ca/dbdesign01/chapter/chapter-12-normalization
[Accessed 5 July 2022].

Candidate Checklist

Please use the following checklist to ensure that your work is ready for submission.

Have you read the NCC Education documents 'What is Academic Misconduct? Guidance for Candidates' and 'Avoiding Plagiarism and Collusion: Guidance for Candidates' and ensured that you have acknowledge all the sources that you have used in your work?	✓
Have you completed the 'Statement and Confirmation of Own Work' form and attached it to your assignment? You must do this.	✓
Have you ensured that your work has not gone over or under the recommended word count by more than 10%?	✓
Have you ensured that your work does not contain viruses and can be run directly?	✓