Assignment 2

CP612 Data Management & Analysis

Faculty of Science - Masters of Applied Computing

Instructor: Kaiyu Li

Group Members:

Arya Vivekanand Prabhu (245840540)

Tonsia Treesa Thomas (245839190)

Shri Gomathivalli Thangappatham (245839690)

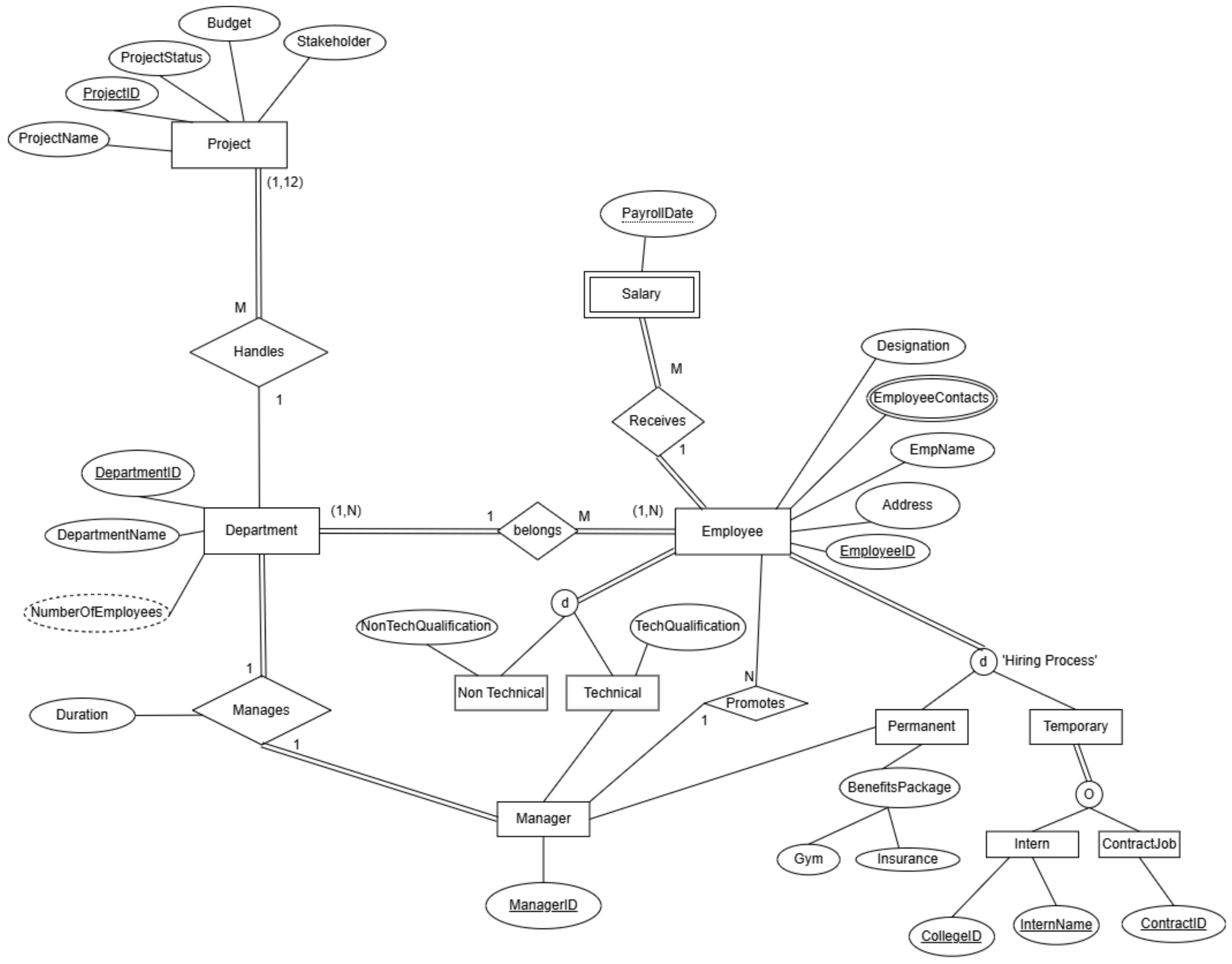
Number of words in document ~ 1841

Executive Summary

This study is an attempt to highlight the current structure used within an IT company, Stream Data Systems for employee management and project development, and present it as a generic and universal solution.

Table of Content

Conceptual Data Model



Relational Model (4 Pages)

Multi Valued and Variable Attributes

Employee has a multi valued attribute called “EmployeeContacts”. We will convert this by creating a new table named “EmployeeContacts” and use EmployeeID as a foreign key. We will repeat the same all multivalued or multivariable attributes

Relationship: Handles

We will convert this relationship using the “Foreign Key Approach for 1:1 Binary relations”. We use DeptID as a foreign key for Project.

Specialization: Hiring Process & Temporary Overlap

To convert the disjoint Employee specialization we will follow the example on Part 2-2 Page 34. All the disjoint entity’s attributes will be added to the table employees. The overlap between Intern and ContractJob will make use of two flags “IFlag” and “CFlag” respectively. Benefits Package will be a separate table with EmployeeID as a foreign key. Intern and ContractJob details will also be stored in a separate table and be referred to as foreign keys.

Employee Receives Salary

Here every Employee must receive exactly one Salary and every Salary is associated with one Employee. Now it's represented as, the Employee and Salary entities can be merged into a single table. The primary key of the Employee table(EmployeeID) is used as the primary key in the merged table which ensures total participation.

Employee Belongs to Department

Here many Employees can belong to one Department, and every Employee must be associated with exactly one Department. So for this, the Employee entity has a foreign key (DepartmentID) referencing the Department entity. And the total participation ensures that every employee must be assigned to a department.

Disjoint Specialization: Tech/Non-Tech Qualification

Here we use the ‘Single Relation with One Type Attribute’ approach by creating a single table with all attributes Employee and the subclasses [Technical and Non-Technical] along with the discriminating attribute [EmployeeType] to help us distinguish between the subclasses [Technical or Non-Technical]. The primary key of the Employee (EmployeeID) is used as the primary key for the table.

Foreign Key Approach for 1:1 Binary Relationships for Department and Manager

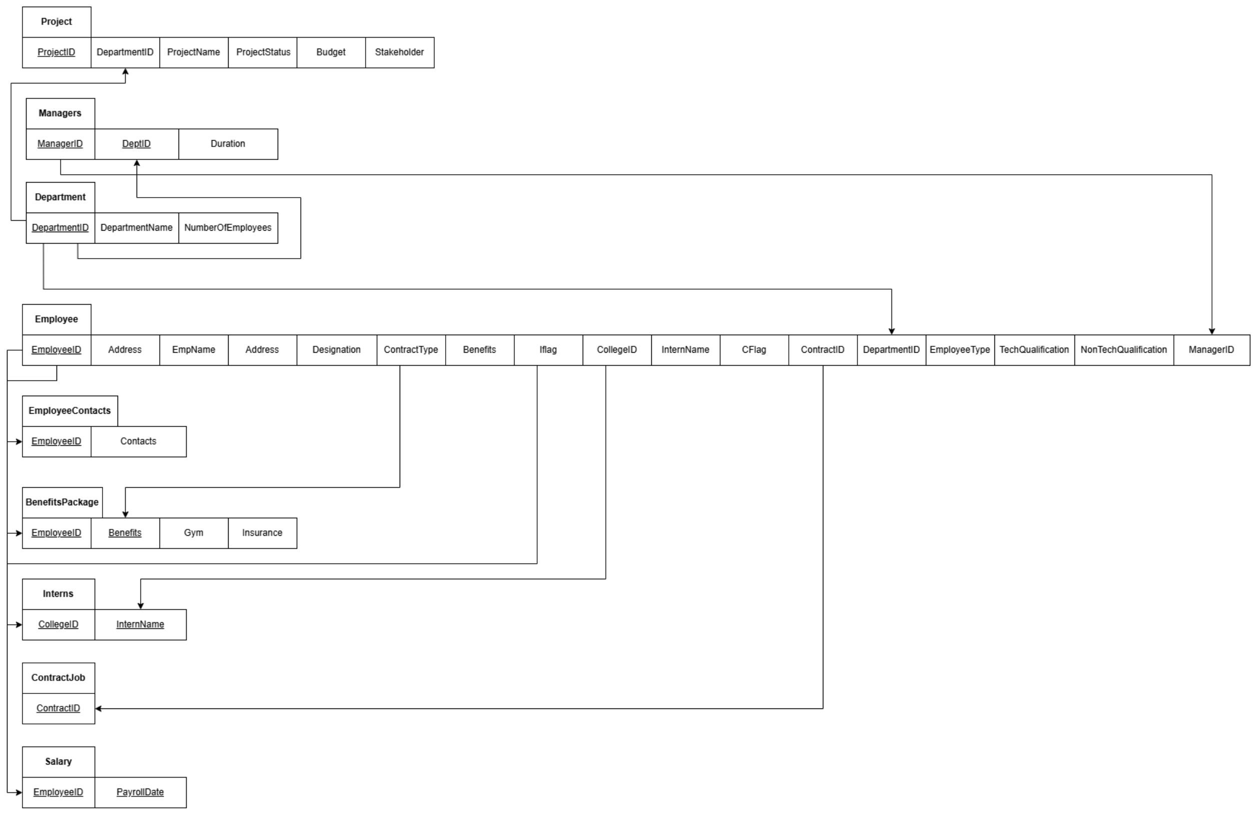
Created one relation for each Entity and added foreign key to reduce the nulls and shifted the relationship attribute to one of the same relation having the foreign key

Mapping 1:N Binary Relationship Types for manager and employee

Include the primary key of the entity type that represents the many-side of the relationship as a foreign key in the entity that represents the one-side of the relationship.

Final Model

Provided below is our final relational model.



Data Dictionary

Using the following legend:

- PK - Primary Key
- FK - Foreign Key
- NN - Not Null
- U - Unique

Employees

Attribute Name	Domain	Constr ains	Meaning	Example Values
EmployeeID	Integer	PK	A unique number assigned to identify each employee in	1001, 1025, 2048

Attribute Name	Domain	Constr ains	Meaning	Example Values
		U NN	the company.	
Designation	String	NN	The job title or role the employee holds in the organization.	Software Engineer, Manager
EmpName	String	NN	The full name of the employee.	Davis Joe, Tomi P
Address	String		The home or office address of the employee.	123 Elm St, Springfield, IL, 456 Oak Rd
ContractType	String	Only ‘P’ or ‘T’	Flag to understand whether the contract is Permanent or Temporary	P T
Benefits	String	FK NN	Describes the benefits package (Foreign key referencing table BenefitsPackage)	Gold New Employee Vista
Iflag	Bool	NN	Flag to highlight whether Employee is an intern	T F
CollegeID	Int	FK NN U	ID of the Intern's college (Foreign key referencing table Interns)	12345 65535
InternName	String	FK NN	Name of the Intern's (Foreign key referencing	Mike Mark

Attribute Name	Domain	Constr ains	Meaning	Example Values
			table Interns)	Arya
Cflag	Bool	NN	Flag to highlight whether Employee is a contract job employee	T F
ContractID	String	FK U NN	A unique identifier only for temporary intern/contract employees. (Foreign Key referencing table Contract Job)	CON56789, INT2456
DepartmentID	String	FK	A unique identifier for departments (Foreign Key referencing table Department)	R1 C54
EmployeeType	String	NN Only 'Tech' or 'NonTech'	Flag highlighting whether the Employee is Technical or Non-Technical	Tech NonTech
NonTechQualification	String	NN	Educational Qualification of the employees working in the Non Technical Background	BBA, B.Com
TechQualification	String	NN	Educational Qualification of the employees working in the Technical Background	B.E, B.Tech, MAC
ManagerID	String	FK U	Unique identifier for the manager (Foreign Key referencing	M123 B238

Attribute Name	Domain	Constrains	Meaning	Example Values
		NN	table Manager)	

Employee Contacts

EmployeeID	Integer	FK,PK U NN	A unique number assigned to identify each employee in the company.	1001, 1025, 2048
EmployeeContacts	String	NN	The contact details (phone numbers or emails) of the employee.	123-456-7890, 987-654-3210, emp@company.com

Manager

ManagerID	String	PK U NN	Unique identifier for the manager	M123 B238
DepartmentID	String	FK	A unique identifier for departments (Foreign Key referencing table Department)	R1 C54
Duration	String		Represents the time period during which the manager oversees the department.	"2020-2023", "Jan 2019 - Dec 2021"

Department

DepartmentID	String	PK U NN	A unique identifier for departments	R1 C54
DepartmentName	String	U NN	The name of the department	“Research Team 2”, “HR”
NumberOfEmployees	Int		Number of employees in that department	2 43

Salary

EmployeeID	Integer	FK,PK U NN	A unique number assigned to identify each employee in the company. (Foreign Key referencing table Employee)	1001, 1025, 2048
PayrollDate	Date		Date when salaries are paid to employees	2025-04-31

Contract Job

ContractID	String	PK U NN	A unique identifier only for temporary intern/contract employees.	CON56789, INT2456

Interns

InternName	String	PK U	Name of the intern that serves as a partial key	Arya Gomathi

		NN		
CollegeID	String	PK U NN	Unique identifier for colleges that interns belong to	T732 Y238

Benefits Package

EmployeeID	Integer	FK,PK U NN	A unique number assigned to identify each employee in the company. (Foreign Key referencing table Employee)	1001, 1025, 2048
BenefitsPackage	Array		A list of benefits offered to the employee	[TRUE, FALSE] [FALSE, FALSE]
Insurance	Boolean		Whether the employee is covered under life and health insurance	TRUE FALSE
Gym	Boolean		Whether the employee is given a benefit of discounted gym access.	TRUE FALSE

Projects

Stakeholder	String	NN	Name of the stakeholder of a project	Mountain Corp. SDS
DepartmentID	String	FK	A unique identifier for departments (Foreign Key referencing table Department)	R1 C54

ProjectID	String	PK NN U CHECK(Projects <=12)	A unique identifier to identify each project given by the shareholder.	Wal23, Day3
ProjectName	String	U NN	The name given to the project for general reference within the company	“Website for Olive”, “Internal System Upgrade Version 5”
ProjectStatus	String		An indicator for the project’s current status	Complete, On Hold
Budget	Float	NN	The project’s current cost in Millions	2.40, 4.67

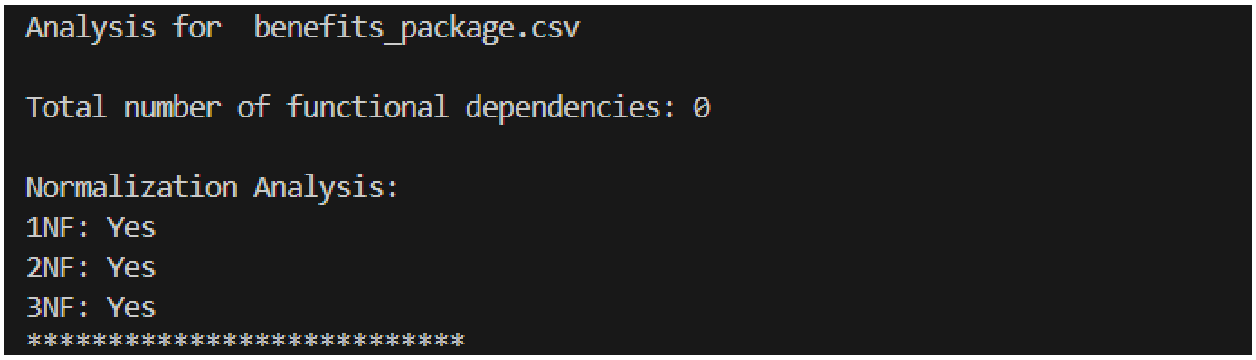
Synthetic Data

In order to generate 100 rows for our schema we could not use an AI bot since the request is too intensive. Instead we use the *Faker* library from python.
Source code can be found submitted as fake.py with all the data stored in folder ‘data’ as csv files.

Normalization Analysis

Lets dive into the level of normalization followed by the table’s structure and analyze the conditions for 1NF, 2NF, and 3NF and we identify any normalization issues.
We conducted the analysis through the brute force method, where we checked each functional dependency to see if the table meets the rules for each normal form. We generated all possible combinations for functional dependencies using *combinations* python library

Analysis for Table **benefits_package**:



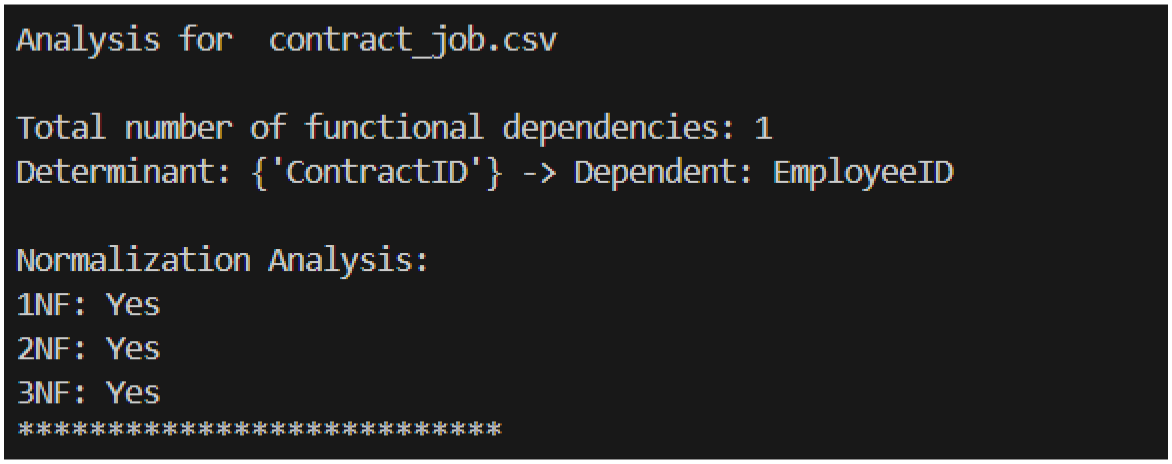
```
Analysis for  benefits_package.csv

Total number of functional dependencies: 0

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: Yes
***************************
```

The table follows all levels of normalization. There are no functional dependencies so the table does not have redundancy, partial dependencies and transitive dependencies.
The table satisfies 1NF, 2NF, and 3NF.

Analysis for Table **contract_job:**



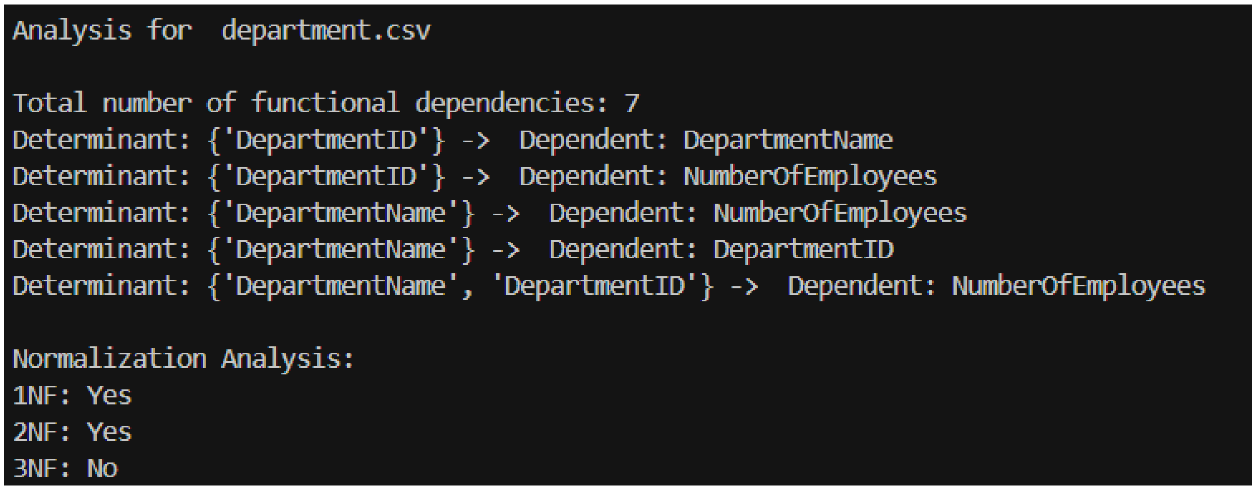
```
Analysis for  contract_job.csv

Total number of functional dependencies: 1
Determinant: {'ContractID'} -> Dependent: EmployeeID

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: Yes
***************************
```

The table follows all levels of normalization. With the functional dependency ContractID→EmployeeID, the table meets the criteria for 1NF, 2NF and 3NF. There are no partial or transitive dependencies.

Analysis for Table **department.csv:**



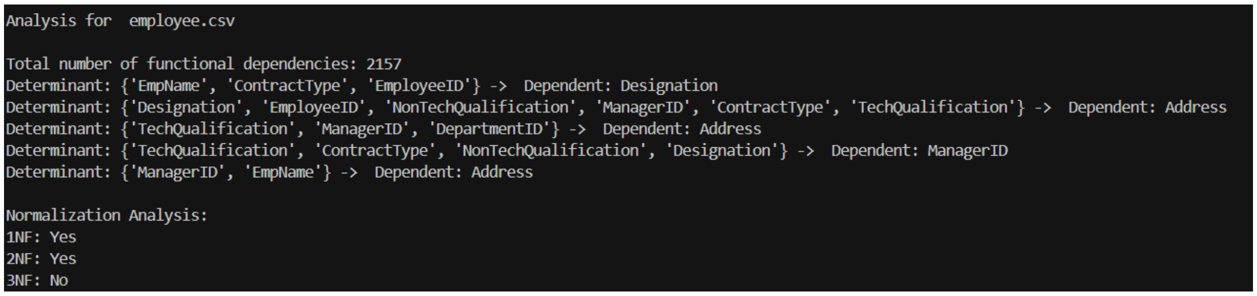
```
Analysis for  department.csv

Total number of functional dependencies: 7
Determinant: {'DepartmentID'} ->  Dependent: DepartmentName
Determinant: {'DepartmentID'} ->  Dependent: NumberOfEmployees
Determinant: {'DepartmentName'} ->  Dependent: NumberOfEmployees
Determinant: {'DepartmentName'} ->  Dependent: DepartmentID
Determinant: {'DepartmentName', 'DepartmentID'} ->  Dependent: NumberOfEmployees

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: No
```

The table follows 1NF and 2NF but does not meet the criteria for 3NF due to transitive dependencies. Here, DepartmentName→DepartmentID→NumberOfEmployees creates a transitive dependency that violates 3NF rules.

Analysis for Table **employee.csv:**



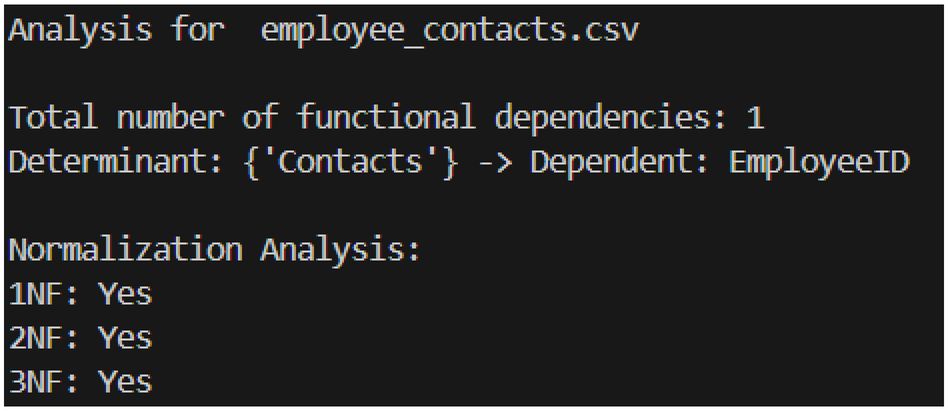
```
Analysis for  employee.csv

Total number of functional dependencies: 2157
Determinant: {'EmpName', 'ContractType', 'EmployeeID'} ->  Dependent: Designation
Determinant: {'Designation', 'EmployeeID', 'NonTechQualification', 'ManagerID', 'ContractType', 'TechQualification'} ->  Dependent: Address
Determinant: {'TechQualification', 'ManagerID', 'DepartmentID'} ->  Dependent: Address
Determinant: {'TechQualification', 'ContractType', 'NonTechQualification', 'Designation'} ->  Dependent: ManagerID
Determinant: {'ManagerID', 'EmpName'} ->  Dependent: Address

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: No
```

The table follows 1NF and 2NF but does not meet the criteria for 3NF due to transitive dependencies. Here, ManagerID→EmpName→Address creates a transitive dependency that violates 3NF rules.

Analysis for Table **employee_contacts.csv:**



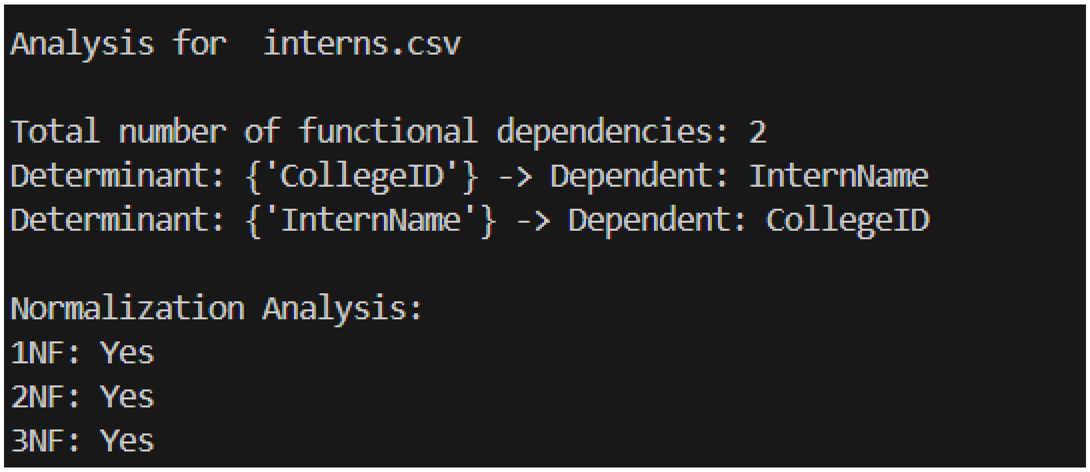
```
Analysis for  employee_contacts.csv

Total number of functional dependencies: 1
Determinant: {'Contacts'} -> Dependent: EmployeeID

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: Yes
```

The table follows all levels of normalization. With the functional dependency ContractID→EmployeeID, the table meets the criteria for 1NF, 2NF and 3NF. There are no partial or transitive dependencies.

Analysis for Table **interns.csv:**



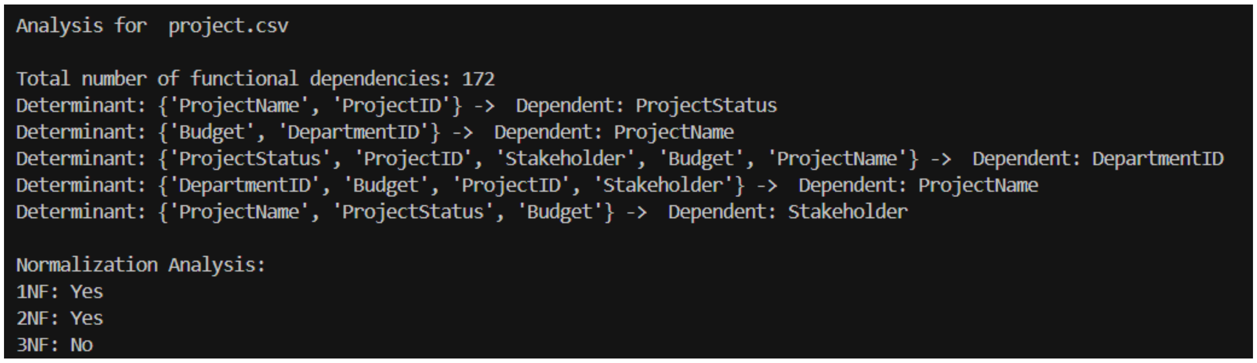
```
Analysis for  interns.csv

Total number of functional dependencies: 2
Determinant: {'CollegeID'} -> Dependent: InternName
Determinant: {'InternName'} -> Dependent: CollegeID

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: Yes
```

The table follows all levels of normalization. With the functional dependency CollegeID→InternName, InternName→CollegeID where the attributes are mutually dependent, indicating a one-to-one relationship. The table meets the criteria for 1NF, 2NF and 3NF. There are no partial or transitive dependencies.

Analysis for Table **project.csv:**



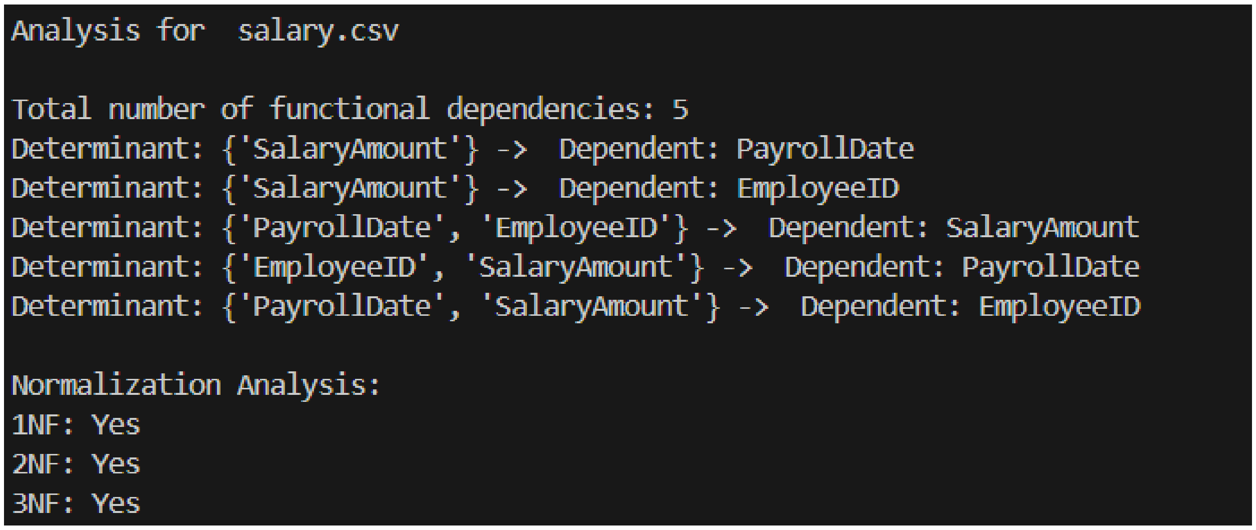
```
Analysis for  project.csv

Total number of functional dependencies: 172
Determinant: {'ProjectName', 'ProjectID'} ->  Dependent: ProjectStatus
Determinant: {'Budget', 'DepartmentID'} ->  Dependent: ProjectName
Determinant: {'ProjectStatus', 'ProjectID', 'Stakeholder', 'Budget', 'ProjectName'} ->  Dependent: DepartmentID
Determinant: {'DepartmentID', 'Budget', 'ProjectID', 'Stakeholder'} ->  Dependent: ProjectName
Determinant: {'ProjectName', 'ProjectStatus', 'Budget'} ->  Dependent: Stakeholder

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: No
```

The table follows 1NF and 2NF but does not meet the criteria for 3NF due to transitive dependencies. Here, ProjectName→ProjectStatus→Stakeholder creates a transitive dependency that violates 3NF rules.

Analysis for Table **salary.csv:**



```
Analysis for  salary.csv

Total number of functional dependencies: 5
Determinant: {'SalaryAmount'} ->  Dependent: PayrollDate
Determinant: {'SalaryAmount'} ->  Dependent: EmployeeID
Determinant: {'PayrollDate', 'EmployeeID'} ->  Dependent: SalaryAmount
Determinant: {'EmployeeID', 'SalaryAmount'} ->  Dependent: PayrollDate
Determinant: {'PayrollDate', 'SalaryAmount'} ->  Dependent: EmployeeID

Normalization Analysis:
1NF: Yes
2NF: Yes
3NF: Yes
```

The table follows all levels of normalization. With the functional dependencies there are no partial or transitive dependencies. The table meets the criteria for 1NF, 2NF and 3NF.

Individual Contributions

This assignment was completed as a collaborative effort, with each member contributing equally. As a result there are no tasks that were completed individually. However, we can highlight the overall effort put in by each one of us:

Gomathi: Helped with the mapping of entities, attributes, and relationships from the conceptual model to a relational model also provided justifications for her choices. Specified the data types, constraints, primary keys and foreign keys for all tables. Came up with the normalization analysis and provided an explanation for 1NF, 2NF, and 3NF along with their functional dependencies.

Tonsia: Contributed by helping with creating the relational diagram also by writing the source code for finding functional dependencies from the synthetic datasets. Help with organizing and structure the report.

Arya: Wrote the program in Python to generate synthetic data for each table based on the relational model and ensured that the data aligns with the data type definitions and constraints. Prepared the data dictionary by listing all attributes, data types and its constraints.