ASSIGNMENT-1

1.The Beautiful House Case Study

1.1 The main advantages of using DBMS for managing the data of beautiful house are:

a) *Data sharing using DBMS i.e., the Beautiful House can share the data across multiple offices. The data is stored in a single location and it can be used widely and shared to all the branches.*

b) *Through the Centralized database system, it can be easily maintained, and this is the one of the greatest benefits that we will be able to access and store the essential information from the centralized location. Manually finding the exact information is a tedious process and with the efficient system can be able to find the data easily and business decisions can be made quickly.*

c) *DBMS helps us to maintain data integrity. Which implies the accuracy and consistency of data is maintained in the database. There will be a high chance of data inconsistencies as there will be several branches in Beautiful House that manipulate and access the same data. So is crucial to maintain Data Integrity.*

1.2 3 Classes of objects – Offices, Staff and Properties.

The Beautiful House data can be represented in the form of database object(tables) whereas the classes of the objects are the unique defined with its set of attributes. For the given case study, the three classes of objects are:

a) **Offices:** *The offices table consists of the following details:*

1. ***Branch_Number**: A unique branch number should be assigned to each branch office.*
2. ***Address**: This field consists of the address i.e., Street, Suburb, and Postcode of the branch office.*
   ***Street**: This field consists of the street.*
   ***Suburb**: Suburb in which the Office is located.*
   ***Postcode**: Postcode of every branch office.*
3. ***Telephone_number:** This field will be filled with maximum of 3 numbers for each branch.*
   It is attributed as,
   ***Telephone1:** for the 1st number.*
   ***Telephone2:** for the 2nd number.*
   ***Telephone3:** for the 3rd number.*
4. ***Manager_Name:** Each branch will be managed by a person and the name of the person will be included.*
5. ***Starting_Date:** The date the manager positioned in the branch.*

6. ***Bonus_Payment:*** *Include the bonus payment given based on the performance.*

*b) **Staff:** The staff table consists of the following details:*

1. ***Staff_Number:*** *A unique staff number should be assigned to every member.*
2. ***Name**: Name of the staff number.*
3. ***Address**: Field will be filled with the address of the staff member.*
4. ***Position:*** *The designation of the member whether an agent or supervisor.*
5. ***Salary:*** *The salary of the staff member.*
6. ***Supervisor_Name**: If the staff member is an agent then the Supervisor name will be applicable and should be mentioned.*
7. ***Branch_Number:*** *To represent the details of the branch where the staff member is working.*

*c) **Property:** The property table consists of the following details:*

1. ***Property_Number:*** *Every property offered by the Beautiful House will be assigned with a unique property number.*
2. ***Address**: This field consists of the address i.e., Street, Suburb, and Postcode of the Property.*
   - ***Street**: This field consists of the street*
   - ***Suburb**: Suburb in which the Office is located*
   - ***Postcode**: Postcode of every branch office.*
3. ***Type:*** *Represents the property type.*
4. ***No_of_rooms:*** *The number of rooms present in the available and given for rent.*
5. ***Weekly_Rent:*** *The weekly rent of the given object.*
6. ***Staff Number:*** *The agent handling the property.*

1.3 Relationships among classes of objects.

*1. **One-to-Many (1 - N or 1 - *):** Each record of the table related to one or more records of other table.*

From the given database,

Office / Branch (1) → (has) 'N' no of properties

Represents One – to – Many relation.

*2. **One-to-One (1 - 1):** Each record of the table relates to only one record of other table.*

From the given database,

Staff (1) → (relates to) Office / Branch (1)

Represents Many – to – One relation.

3. ***Many-to-Many (N - N or * - *):*** *Each record of the first table relates to one or more records of second table and vice versa. A many-to-many relation can be represented as two one-to-many relations.*

From the given database,

Properties (N) → (handled by) 'N' number of Staff.

Represents Many – to – Many relation.

1.4 Users of database.

There are different kinds of users for database system. Even though all the users need not use the database necessarily. Based on the case study the different kinds of users for the database system are:

DBA:

- *Responsible for the management of overall database.*
- *To define instances, schemas etc.*

User:

- *Property searching.*
- *Get the details of the staff or office corresponding to the property (If stored in the database).*

Admin:

- *Property insertion or adding a new property.*
- *Property Updating.*
- *Property removal or deleting the property.*
- *Similar operations are applicable on staff and their details also.*

2.Employee Database

The Schema for the Employee Database is given below:
Employees (employee_id, first_name, last_name, phone_number, hire_date, job_id, salary, department_id)

Departments (department_id, department_name, manager_id*, location_id*)

Jobs (job_id, job_title, min_salary, max_salary)

Locations (location_id, street_address, postal_code, city, state_province, country_id*)

Countries (country_id, country_name)

JobHistory (employee_id*, start_date, end_date, job_id*, department_id*)

2.1 Employee - Job association.

On considering the employees table though the job_id column was included, but the column was not referenced as the foreign key. Therefore, the job role i.e., job_title cannot be referenced to the employees table directly.

2.2 Can an Employee have 2 Jobs and work for 2 Departments at the same time?

***Alternative 1** – No, he cannot. Even though the job_id and dept_id are not foreign keys and are not even declared as primary keys he can work in 2 departments and have 2 jobs at the same time. Being a primary key employee_id can have only one entry. So, if he tries storing 2 different jobs in different departments in the database it will not be possible. Even job_id and department_id can be duplicated. Also, we cannot have multiple tuple values.*

***Alternative 2** – As there is no specification for datatype in the schema and neither they are not the children to any class attribute. We can declare the type of the variable as "Varchar2" and then we can insert the values for example say '[22, 10]'. Then we can have 2 jobs and 2 departments at the same time.*

***Alternative 3** – Same as 2.1 as there is no job associated with the employee there will be no job role for an employee even if there is a job_id corresponding to the employee_id. So, he cannot have 2 jobs and 2 departments at the same time.*

2.3 Given queries,

-- INSERT INTO Departments VALUES (2, 'Accounting', 66, 10);

-- INSERT INTO Departments VALUES (2, 'Accounting', 66, 30);

The two statements given to insert the values of accounting department in the department table will not work as the department_ id is considered as the primary key the duplicate values will not be accepted. As, the department_id with 2 already exists the following given insert statements will not work.

2.4 Finding the Manager details.

The request to find all the employment details of department managers can be completed using the given database schema.

```
SELECT e.*
FROM EMPLOYEES e, DEPARTMENTS d
WHERE e.employee_id = d.manager_id;
```

The above specified query extracts all the data from the employee table by verifying the condition given in the WHERE clause. The query execution takes place as shown below:

1. ***FROM:***
   Here in this step it retrieves all the data stored in the tables "employees" and "departments" and corresponds them to the respective objects.
   e – employees
   d – departments
2. ***WHERE:*** *In the second step a condition is specified, and the data is retrieved based on the condition provided.*
   Condition: *Returns the ids where the employee_id matches with the manager_id.*
3. ***SELECT****: Finally, the select statement the selects the data from the given employees table corresponding to the ids returned in the where clause.*

Therefore, the details of all the department managers is obtained.

2.5 -- DELETE from JOBS WHERE job_id = 33;

On executing the given SQL statement the row with job_id 33 will not be deleted from the table JOBS as the job_id is the foreign key to the table JOBHISTORY, without using DELETE CASCADE in the JOBS table the data in the JOBHISTORY will be orphaned. So, instead of DELETE if we use DELETE CASCADE the rows with job_id 33 will also get deleted in the JOBHISTORY table. In the second option without deleting the entire table or rows we can update the values to NULL by using ON UPDATE SET NULL CASCADE.

2.6 Creating the table Department using the CREATE statement:

Creating a table, requires the assurance that a database does not have a duplicate table i.e., the table that is to be created should not have a similar name to that is already present in the database. For large databases containing many tables it becomes difficult for us to check whether the table already exists or not. So, to achieve this, we simply use the drop query to drop the tables in the order of their dependencies (i.e., by considering the constraints like foreign keys).

Syntax:- drop table tablename;

Dropping the table.

```
drop table Departments;
```

Now create the table Departments.

```
Create table Departments (
department_id NUMBER (10),
department_name varchar2(40),
manager_id NUMBER (2),
location_id NUMBER (10),
CONSTRAINT d_pk_id PRIMARY KEY (department_id),
CONSTRAINT mn_id_fk FOREIGN KEY (manager_id)
REFERENCES employees(employee_id),
CONSTRAINT l_id_fk FOREIGN KEY (location_id)
REFERENCES locations(location_id)
);
```

2.7 Creating the table JobHistory using the CREATE statement:

Dropping the table.

```
drop table Departments;
```

Now create the table JobHistory.

```
CREATE TABLE JobHistory (
employee_id NUMBER (10),
start_Date DATE,
end_Date DATE,
job_id NUMBER (10),
department_id NUMBER (10),
CONSTRAINT jh_pk_id PRIMARY KEY (employee_id,
start_Date, end_Date),
CONSTRAINT emp_id_fk FOREIGN KEY (employee_id)
REFERENCES Employees (employee_id),
```

```
CONSTRAINT job_id_fk FOREIGN KEY (job_id)
REFERENCES Jobs(job_id),
CONSTRAINT dept_id_fk FOREIGN KEY
(department_id) REFERENCES Departments
(department_id)
);
```

2.8 *Add a new department in the address "555 Swanston Street".*

Schema -

```
Departments(department_id, department_name,
manager_id*, location_id*)
```

Query -

```
INSERT INTO DEPARTMENTS VALUES (4, 'Property
Services', NULL, 30);
```

- ***department_id*** *is any number say* ***4****.*
- ***department_name - 'Property Services'***
- ***Manager_id*** *is* ***NULL*** *(Since manager is not assigned).*
- ***location_id*** *is* ***30*** *implies the address* ***"555 Swanston Street".***