LAB Manual
PART A
(PART A : TO BE REFERRED BY STUDENTS)

Experiment No.02

A.1 Aim:
Implementation of all dimension tables and fact tables related to case study mentioned in the first experiment.
In this experiment , we write the SQL queries for,
1.Creating the dimension table
2.Creating the Fact table
3. Inserting values in both tables.
4. Displaying the tables.
5.Draw the Fact Constellation Schema.

A.2 Prerequisite:
Refer to the DBMS manual for SQL Commands and ER diagram.
A.3 Outcome:
After successful completion of this experiment students will be able to
- □ Model and create schemas for data warehouses.

A.4 Theory:

Dimensions are categories by which summarized data can be viewed. E.g. A profit summary in a fact table can be viewed by a Time dimension (profit by month, quarter, year), Region dimension (profit by country, state, city), Product dimension (profit for product1, product2).
A fact table is a table that contains summarized numerical and historical data (facts) and a multipart index composed of foreign keys from the primary keys of related dimension tables.
In data warehousing, a dimension is a collection of reference information about a measurable event. These events are known as facts and are stored in a fact table. Dimensions categorize and describe data warehouse facts and measures in ways that support meaningful answers to business questions. They form the very core of dimensional modeling.
Dimension tables are referenced by fact tables using keys. When creating a dimension table in a data warehouse, a system-generated key is used to uniquely identify a row in the dimension. This key is also known as a surrogate key. The surrogate key is used as the primary key in the dimension table. The surrogate key is placed in the fact table and a foreign key is defined between the two tables. When the data is joined, it does so just as any other join within the database.

Algorithm:
CREATION OF OLTP TABLES
1) Customer table

```
SQL>CREATE TABLE customer
  2 ( customer_id     VARCHAR2(10) PRIMARY KEY,
  3 name              VARCHAR2(40) NOT NULL,
  4 addr              VARCHAR2(10) NOT NULL,
  5 dob               DATE,
  6 in_range          NUMBER,
  7 h_owner           NUMBER,
  8 c_owner           NUMBER );
```

The CUSTOMER table is populated using the following sample DML statemen

```
SQL>INSERT INTO CUSTOMER VALUES ('R41', 'Devarsee banerjee' ,Chennai',
   '16-dec-1947' , 1 ,0 ,1);
```

2) Item table

```
SQL>CREATE TABLE item
     2 ( item_id        VARCHAR2(20) PRIMARY KEY,
     3 name             VARCHAR2(20) NOT NULL,
     4 brand            VARCHAR2(20) NOT NULL,
     5 dept             NUMBER,
     6 c_price          NUMBER,
     7 s_price          NUMBER,
     8 stock            NUMBER);
```

The ITEM table is populated using the sample DML statement

```
SQL>INSERT INTO ITEM VALUES ( 'R4CB84', 'talc' ,'ponds', 5 ,28 ,34 , 21);
```

3) Trans table

```
SQL>CREATE TABLE trans
     2 ( transid        VARCHAR2(20) PRIMARY KEY,
     3 custid           VARCHAR2(20) REFERENCES
     4                   customer(customer_id),
     5 datet            DATE,
     6 amt              NUMBER,
     7 branchid         NUMBER(2) );
```

The TRANS table is populated using the following sample DML statement

```
SQL>INSERT INTO TRANS VALUES ('R4T81', 'R4200' ,'9-Jan-2003', 47684, 4);
```

4) Item_sold table

```
SQL> CREATE TABLE item_sold
     2 ( transid      VARCHAR2(20) REFERENCES
trans(transid),
     3 itemid         VARCHAR2(20) REFERENCES
item(item_id),
```

```
SQL>INSERT INTO ITEM_SOLD VALUES ('R4T996', 'R4SP16' ,3);
```

5) Branch table

```
SQL>CREATE TABLE branch
     2 ( branchid     NUMBER(2) PRIMARY KEY ,
     3 street         VARCHAR2(54),
     4 city           VARCHAR2(54),
     5 state          VARCHAR2(54) );
```

The BRANCH table is populated using the following DML statement

```
SQL>INSERT INTO BRANCH VALUES ( 'Prasad', 'Banglore' ,'karnataka' );
```

IMPLEMENTATION OF ENTERPRISE DATAMART

CREATION OF DIMENSION TABLES

1) Customer dimension table

```
SQL> CREATE TABLE customer
  2 ( customer_id   NUMBER   PRIMARY KEY,
  3 name            VARCHAR2(40)  NOT NULL,
  4  addr           VARCHAR2(10)    NOT NULL,
  5  dob            DATE,
  6  in_range       NUMBER,
  7  h_owner        NUMBER,
  8  c_owner        NUMBER);
```

2) Item dimension table

```
SQL> CREATE TABLE item
  2 ( item_id         NUMBER  PRIMARY KEY,
  3  name             VARCHAR2(20)  NOT NULL,
  4 brand             VARCHAR2(20)  NOT NULL,
  5 dept        NUMBER,
  6 c_price           NUMBER,
  7 s_price           NUMBER,
  8 stock             NUMBER);
```

3) Branch dimension table

```
SQL> CREATE TABLE branch
  2 (branchid   NUMBER  PRIMARY KEY,
  3   street      VARCHAR2(54),
  4 city         VARCHAR2(54),
  5 state        VARCHAR2(54));
```

4) Time dimension table

```
SQL> CREATE TABLE time
  2 ( timeid    NUMBER  PRIMARY KEY,
  3 day         VARCHAR2(3),
  4 month       VARCHAR2(4),
  5 year        VARCHAR2(3));
```

CREATION OF FACT TABLE

1) Sales_fact table

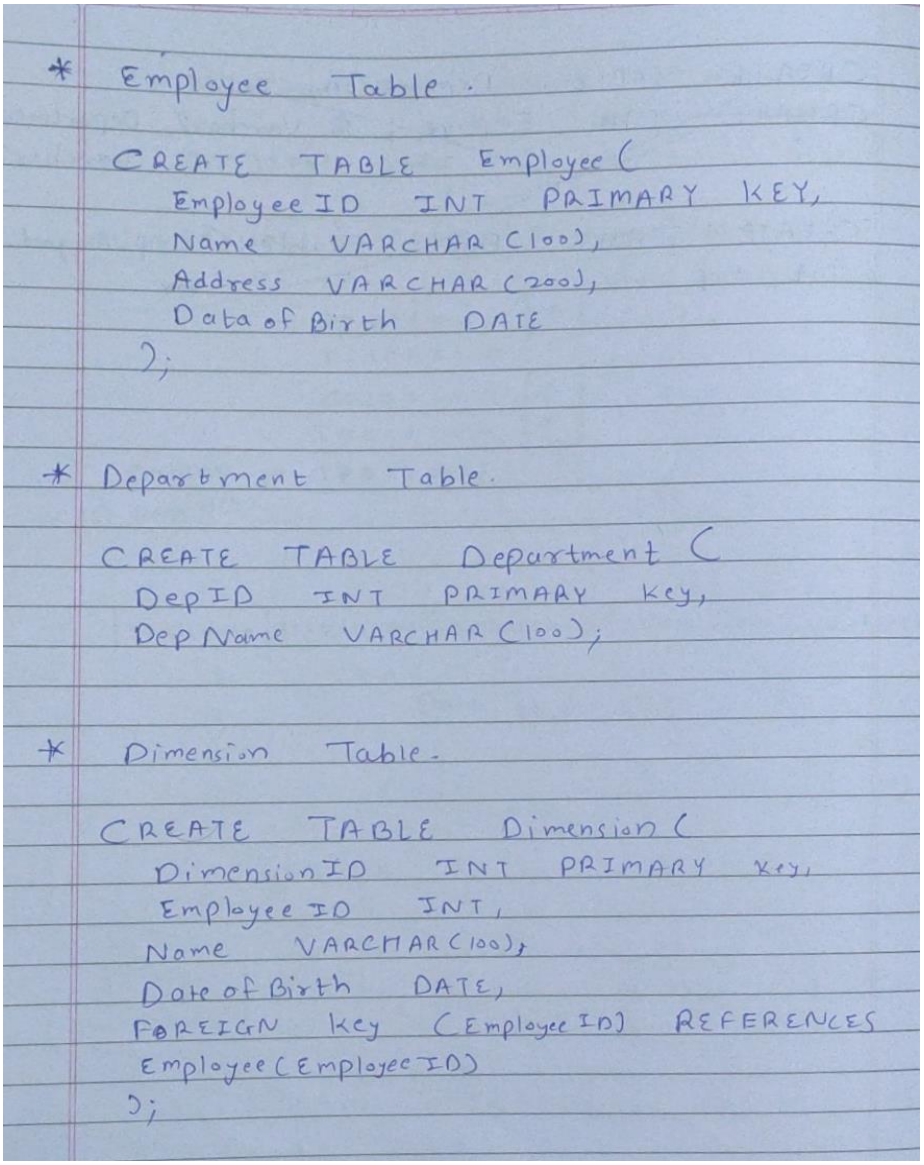
```
SQL> CREATE TABLE sales_fact
  2  (custid          NUMBER  REFERENCES  customer(customer_id),
  3   itemid          NUMBER  REFERENCES  item(item_id),
  4 branchid          NUMBER    REFERENCES branch(branchid) ,
  5  timeid           NUMBER     REFERENCES  time(timeid)
  6 qty               NUMBER,
  7 profit            NUMBER) ;
```

PART B :
(TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll. No. A12	Name: Sufiyan Khan
Class TE. AI & DS	Batch: A1
Experiment No.: 2	Date of Experiment: 25/07/2023
Date of Submission:31/07/2023	Grade:

B.1 Input and Output:
Input :
SQL commands/script



* Employee Assignment Fact Table:

```
CREATE TABLE EmployeeAssignment (
AssignmentID INT PRIMARY Key,
EmployeeID INT,
DepartmentID INT,
StartDate DATE,
EndDate DATE,
FOREIGN Key (EmployeeID) REFERENCES
Employee (EmployeeID),
FOREIGN Key (DepartmentID) REFERENCES
Department (DepartmentID);
```

* Inserting Values

```
INSERT INTO Employee € Values
( 1 , Somesh , Karjat , 2002-12-25 );
INSERT INTO Employee Values
( 2 , Yash , Alibaug , 2004-01-04 );

INSERT INTO Department values
( 11 , AI );
INSERT INTO Department values
( 12 , Comp );
```

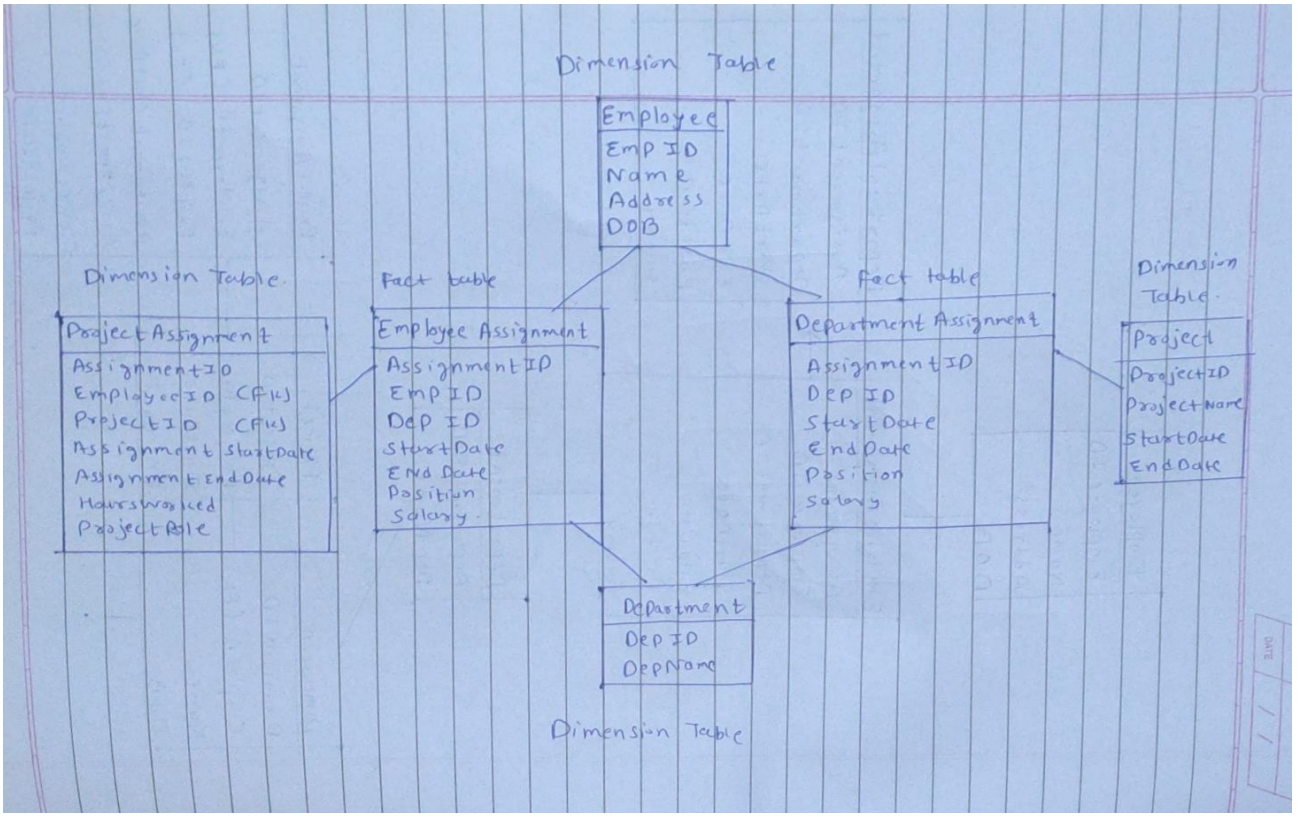
```
INSERT INTO Dimension
SELECT EmpID, Name, DOB
From Employee;

INSERT INTO Fact ~~values~~ (AssignmentID,
Start Date, END Date ) Values
( 123 , 2-20 , 3-20 );
```

* Displaying the Tables.

Select * FRom ~~Empty~~ Dimension Table
AND Fact Table.

Output: Two dimensional Tables created after firing above sql commands.



B.2 Observations and learning:

Observation: The Fact Constellation Schema is a comprehensive and complex data modeling approach that allows for flexible relationships between multiple fact tables and their associated dimension tables. It provides a powerful representation for complex analytical scenarios in data warehousing and business intelligence environments. The schema design can handle various business processes and their interconnections efficiently.

Learning:

- *Flexibility in Relationships: The Fact Constellation Schema demonstrates how multiple fact tables can share common dimension tables, enabling different fact tables to be related to each other. This flexibility allows for better representation of complex business scenarios and supports comprehensive analysis.*
- *Extensibility and Scalability: The schema's design accommodates the addition of new fact tables and dimension tables as the data warehousing needs evolve. This adaptability ensures that the schema can scale and grow to handle increasing data volumes and changing business requirements.*
- *SQL Commands for Table Creation: The SQL commands to create tables (**CREATE TABLE**) are fundamental in setting up the database schema. In the provided examples, the commands include column definitions, primary keys, and foreign keys to establish relationships between tables.*

B.3 Conclusion:

In conclusion, the Fact Constellation Schema is a powerful data modeling technique that accommodates complex relationships between multiple fact tables and dimension tables. Its versatility makes it suitable for handling intricate analytical scenarios in data warehousing and business intelligence

B.4 Question of Curiosity
(To be answered by student based on the practical performed and learning/observations)

Q1: What are the differences between Dimension table and the fact table?

	Dimension Table	***Fact Table***
Purpose	*Stores descriptive attributes for context*	*Stores quantitative measures or metrics*
Data Content	*Contains textual or descriptive data*	*Contains numerical data (measures)*
Cardinality	*Higher cardinality (multiple distinct values)*	*Lower cardinality (aggregated values)*
Normalization	*Usually normalized to minimize redundancy*	*Often denormalized for improved query performance*
Structure	*Flat structure with a single dimension's attributes*	*Long and narrow structure with multiple measures*
Relationships	*Connected to the fact table via foreign keys*	*Contains foreign keys referencing dimension tables*

Q2: Explain Primary Keys, Surrogate Keys & Foreign Keys with an example.

Primary Key: A Primary Key is a unique identifier for each record in a database table. It ensures that each row in the table can be uniquely identified and serves as a reference for maintaining data integrity and enabling efficient data retrieval. A primary key must have the following properties: uniqueness, non-null values, and immutability (should not change over time).
Example: Consider a table named "Employees" with the following columns:

- *EmployeeID (Primary Key)*
- *EmployeeName*
- *Department*
- *Salary*

The "EmployeeID" column is designated as the primary key. Each employee in the table will have a unique EmployeeID, which serves as the identifier for that specific record.
Surrogate Key: A Surrogate Key is an artificial or system-generated unique identifier that serves as the primary key of a table. It is often introduced when the natural keys (existing unique identifiers) of the data are unsuitable for use as the primary key due to issues like complexity, changes, or potential for duplicates. Surrogate keys provide a stable, simple, and unique way to identify records.
Example: Let's consider the same "Employees" table as before but without a natural key that can serve as the primary key. In this case, we can introduce a new column called "EmployeeID" as an auto-incrementing or system-generated value to act as a surrogate key.

- *EmployeeID (Surrogate Key - auto-incremented value)*
- *EmployeeName*
- *Department*
- *Salary*

The "EmployeeID" column is now used as a surrogate key, ensuring each employee record has a unique identifier.
Foreign Key: A Foreign Key is a column or set of columns in a table that refers to the primary key of another table. It establishes a relationship between two tables, enforcing referential integrity and ensuring that data consistency is maintained between related tables. The values in

the foreign key column must correspond to existing values in the primary key of the referenced table or be set to NULL.
Example: Let's introduce a new table called "Departments" with the following columns:

- *DepartmentID (Primary Key)*
- *DepartmentName*

Now, in the "Employees" table, we add a column "DepartmentID" as a foreign key that references the "DepartmentID" column in the "Departments" table.

- *EmployeeID (Primary Key)*
- *EmployeeName*
- *DepartmentID (Foreign Key referencing Departments.DepartmentID)*
- *Salary*

In this example, the "DepartmentID" column in the "Employees" table serves as a foreign key, creating a relationship between the "Employees" table and the "Departments" table. The foreign key ensures that an employee's department ID exists in the "Departments" table, maintaining data integrity and allowing queries to retrieve related information across both tables.