Veermata Jijabai Technological Institute, Mumbai 400019

Experiment No.: 01

Aim: To perform a multidimensional data model using SQL queries. e.g., snowflake, star and fact constellation schema.

Name: Kiran K Patil

Enrolment No.: 211070904

Branch: Computer Engineering

Batch: D

Theory :

Multidimensional Data Modeling Multidimensional Schema is especially designed to model data warehouse systems. The schemas are designed to address the unique needs of very large databases designed for the analytical purpose (OLAP).

Schema

- Schema gives the logical description of the entire database.
- It gives the details about the constraints placed on the table , how the key values are linked between the different tables.

Types of Data Warehouse Schema

1. Star schema
2. Snowflake schema
3. Fact Constellation Schema

1. Star schema

- Star schema is a mature modelling approach widely adopted by relational data warehouses. It requires modelers to classify their model tables as either dimension or fact.
- Dimension tables describe business entities—the things you model. Entities can include products, people, places, and concepts including time itself. The most consistent table you'll find in a star schema is a date dimension table. A dimension table contains a key column (or columns) that acts as a unique identifier, and descriptive columns.
- Fact tables store observations or events, and can be sales orders, stock balances, exchange rates, temperatures, etc. A fact table contains dimension key columns that relate to dimension tables, and numeric measure columns. The dimension key columns determine the dimensionality of a fact table, while the dimension key values determine the granularity of a fact table.
- For example, consider a fact table designed to store sale targets that has two-dimension key columns Date and Product Key. It's easy to understand that the table has two dimensions. The granularity, however, can't be determined without considering the dimension key values. In this example, consider that the values stored in the Date column are the first day of each month. In this case, the granularity is at month-product level.
- Generally, dimension tables contain a relatively small number of rows. Fact tables, on the other hand, can contain a very large number of rows and continue to grow over time.
- The star schema is intensely suitable for data warehouse database design because of the following features:

    - It creates a DE-normalized database that can quickly provide query responses.

- It provides a flexible design that can be changed easily or added to throughout the development cycle, and as the database grows.
- It provides a parallel in design to how end-users typically think of and use the data.
- It reduces the complexity of metadata for both developers and end-users.

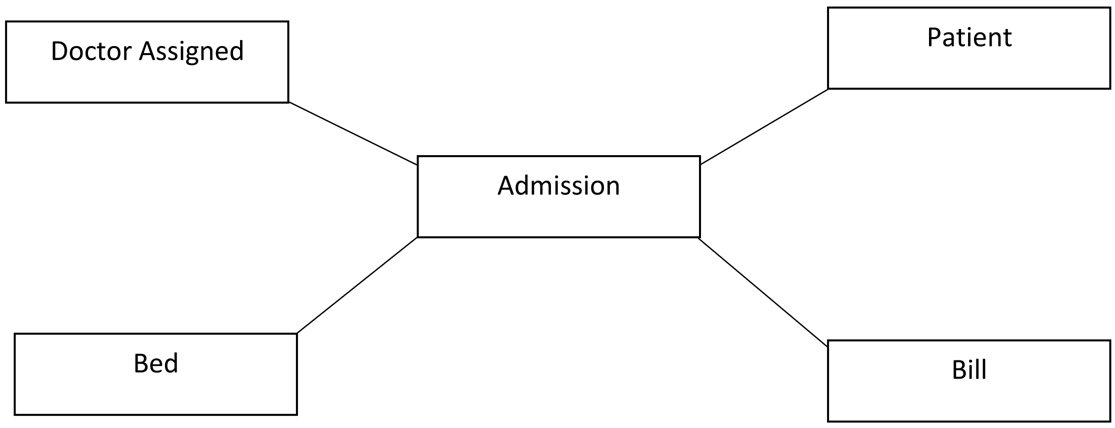
Advantages of Star Schema

1. **Query Performance**: A star schema database has a limited number of table and clear join paths, the query run faster than they do against OLTP systems. Small single-table queries, frequently of a dimension table, are almost instantaneous. Large join queries that contain multiple tables takes only seconds or minutes to run. In a star schema database design, the dimension is connected only through the central fact table. When the two-dimension table is used in a query, only one join path, intersecting the fact tables, exist between those two tables. This design feature enforces authentic and consistent query results.
2. **Load performance and administration:** Structural simplicity also decreases the time required to load large batches of record into a star schema database. By describing facts and dimensions and separating them into the various table, the impact of a load structure is reduced. Dimension table can be populated once and occasionally refreshed. We can add new facts regularly and selectively by appending records to a fact table.
3. **Built-in referential integrity:** A star schema has referential integrity built- in when information is loaded. Referential integrity is enforced because each data in dimensional tables has a unique primary key, and all keys in the fact table are legitimate foreign keys drawn from the dimension table. A record in the fact table which is not related correctly to a dimension cannot be given the correct key value to be retrieved.
4. **Easily Understood**: A star schema is simple to understand and navigate, with dimensions joined only through the fact table. These joins are more significant to the end-user because they represent the fundamental relationship between parts of the underlying business. Customer can also browse dimension table attributes before constructing a query.

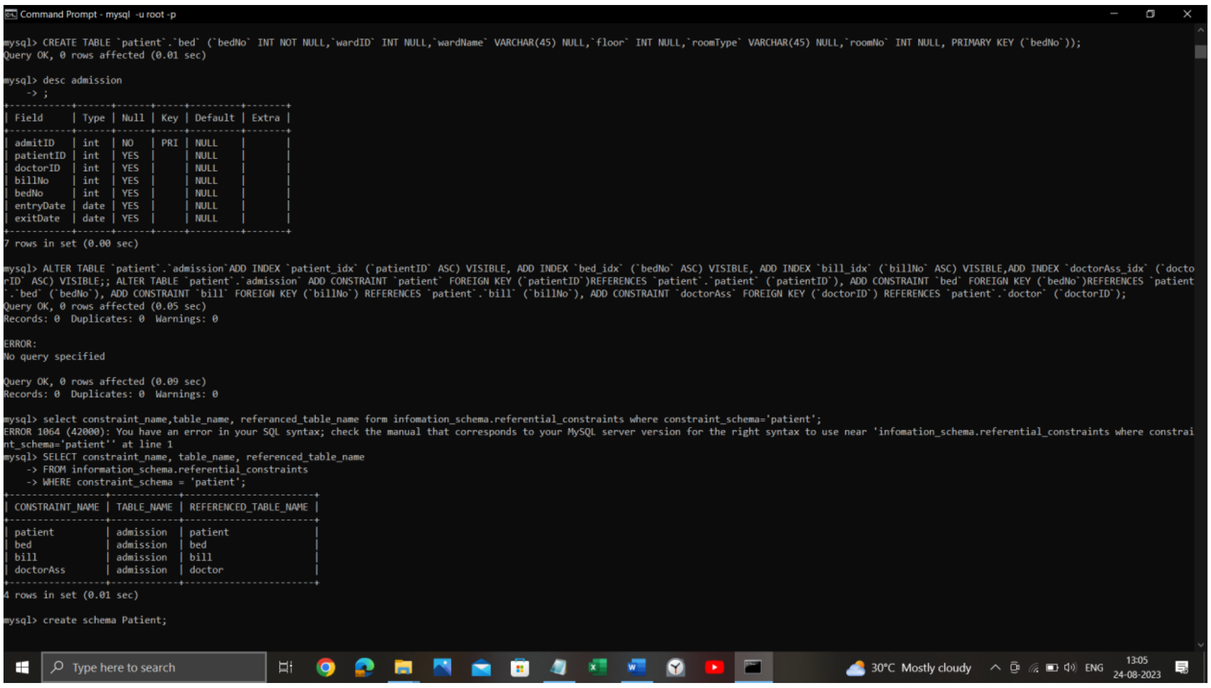
Disadvantage of Star Schema

1. There is some condition which cannot be meet by star schemas like the relationship between the user, and bank account cannot describe as star schema as the relationship between them is many to many.

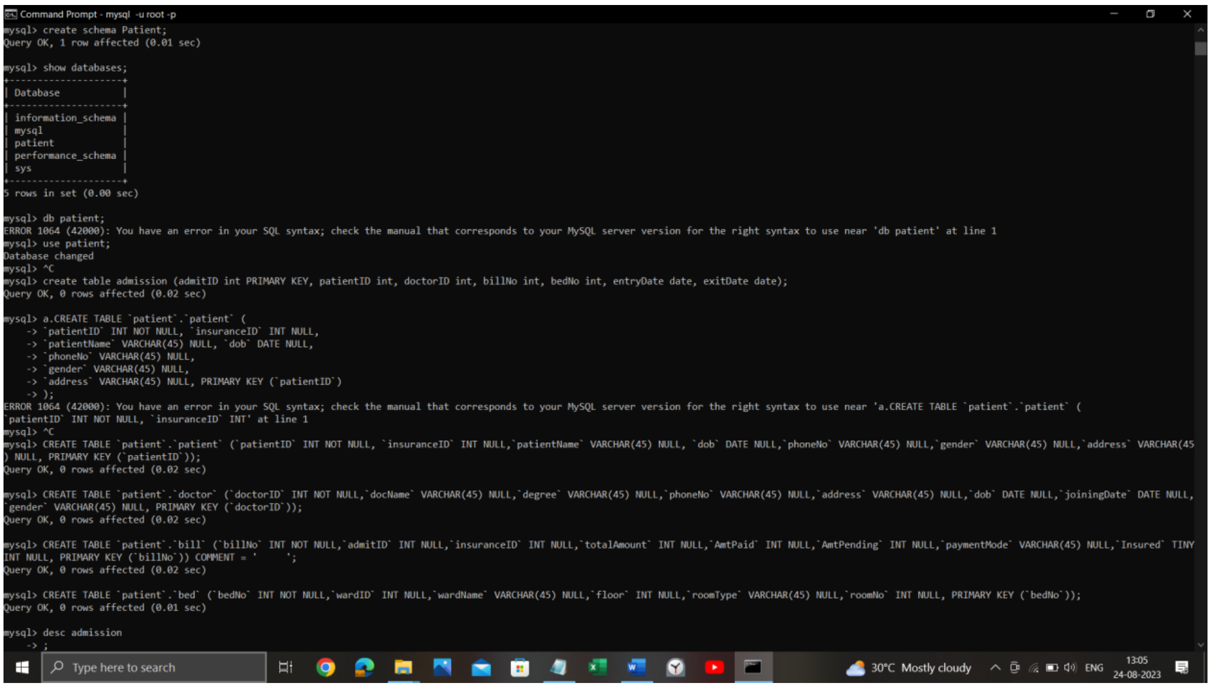
Implementing star schema:



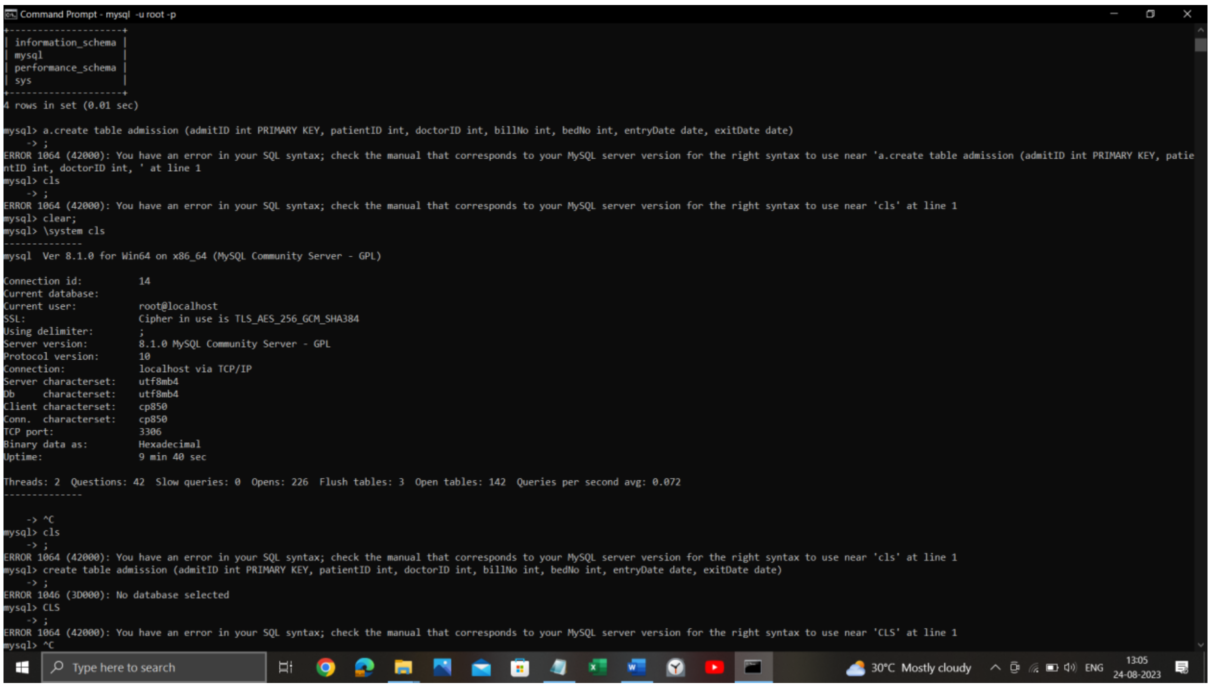
Implementation :



```
mysql> CREATE TABLE `patient`.`bed` (`bedNo` INT NOT NULL,`wardID` INT NULL,`wardName` VARCHAR(45) NULL,`floor` INT NULL,`roomType` VARCHAR(45) NULL,`roomNo` INT NULL, PRIMARY KEY (`bedNo`));
Query OK, 0 rows affected (0.01 sec)

mysql> desc admission
    -> ;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| admitID   | int  | NO   | PRI | NULL    |       |
| patientID | int  | YES  |     | NULL    |       |
| doctorID  | int  | YES  |     | NULL    |       |
| billNo    | int  | YES  |     | NULL    |       |
| bedNo     | int  | YES  |     | NULL    |       |
| entryDate | date | YES  |     | NULL    |       |
| exitDate  | date | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> ALTER TABLE `patient`.`admission`ADD INDEX `patient_idx` (`patientID` ASC) VISIBLE, ADD INDEX `bed_idx` (`bedNo` ASC) VISIBLE, ADD INDEX `bill_idx` (`billNo` ASC) VISIBLE,ADD INDEX `doctorAss_idx` (`docto
rID` ASC) VISIBLE;; ALTER TABLE `patient`.`admission` ADD CONSTRAINT `patient` FOREIGN KEY (`patientID`)REFERENCES `patient`.`patient` (`patientID`), ADD CONSTRAINT `bed` FOREIGN KEY (`bedNo`)REFERENCES `patient
`.`bed` (`bedNo`), ADD CONSTRAINT `bill` FOREIGN KEY (`billNo`) REFERENCES `patient`.`bill` (`billNo`), ADD CONSTRAINT `doctorAss` FOREIGN KEY (`doctorID`) REFERENCES `patient`.`doctor` (`doctorID`);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

ERROR:
No query specified

Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select constraint_name,table_name, referanced_table_name form infomation_schema.referential_constraints where constraint_schema='patient';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'infomation_schema.referential_constraints where constrai
nt_schema='patient'' at line 1
mysql> SELECT constraint_name, table_name, referenced_table_name
    -> FROM information_schema.referential_constraints
    -> WHERE constraint_schema = 'patient';
+-----------------+------------+-----------------------+
| CONSTRAINT_NAME | TABLE_NAME | REFERENCED_TABLE_NAME |
+-----------------+------------+-----------------------+
| patient         | admission  | patient               |
| bed             | admission  | bed                   |
| bill            | admission  | bill                  |
| doctorAss       | admission  | doctor                |
+-----------------+------------+-----------------------+
4 rows in set (0.01 sec)

mysql> create schema Patient;
```



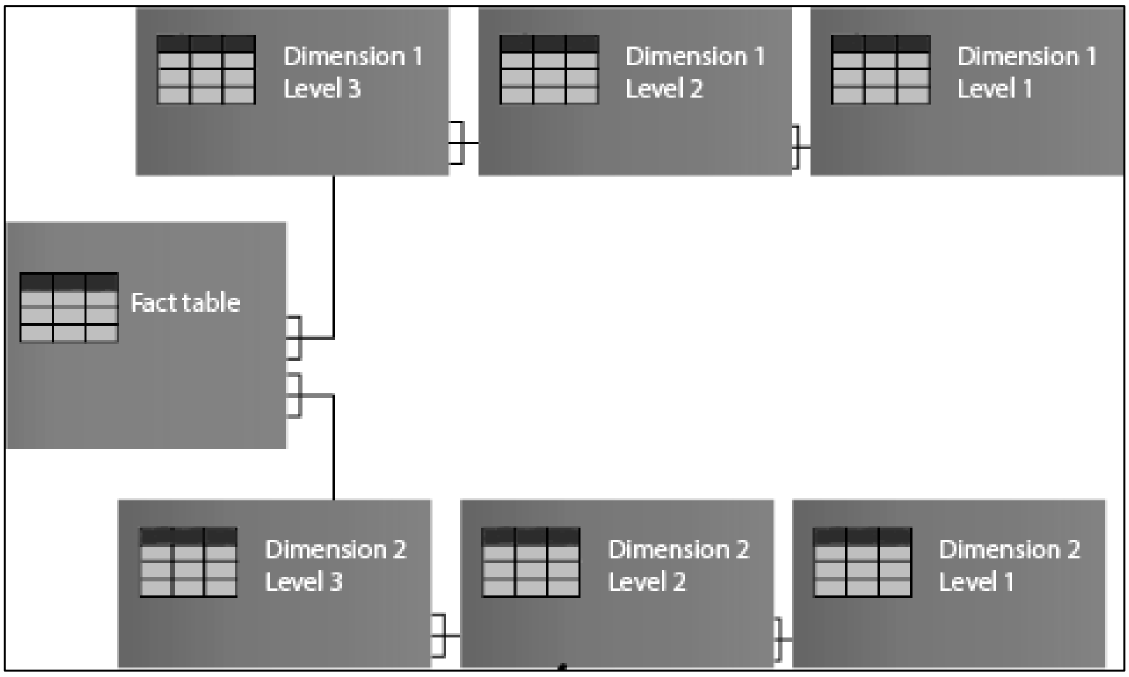
```
mysql> create schema Patient;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| patient            |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> db patient;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'db patient' at line 1
mysql> use patient;
Database changed
mysql> ^C
mysql> create table admission (admitID int PRIMARY KEY, patientID int, doctorID int, billNo int, bedNo int, entryDate date, exitDate date);
Query OK, 0 rows affected (0.02 sec)

mysql> a.CREATE TABLE `patient`.`patient` (
    -> `patientID` INT NOT NULL, `insuranceID` INT NULL,
    -> `patientName` VARCHAR(45) NULL, `dob` DATE NULL,
    -> `phoneNo` VARCHAR(45) NULL,
    -> `gender` VARCHAR(45) NULL,
    -> `address` VARCHAR(45) NULL, PRIMARY KEY (`patientID`)
    -> );
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'a.CREATE TABLE `patient`.`patient` (
`patientID` INT NOT NULL, `insuranceID` INT' at line 1
mysql> ^C
mysql> CREATE TABLE `patient`.`patient` (`patientID` INT NOT NULL, `insuranceID` INT NULL,`patientName` VARCHAR(45) NULL, `dob` DATE NULL,`phoneNo` VARCHAR(45) NULL,`gender` VARCHAR(45) NULL,`address` VARCHAR(45
) NULL, PRIMARY KEY (`patientID`));
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE `patient`.`doctor` (`doctorID` INT NOT NULL,`docName` VARCHAR(45) NULL,`degree` VARCHAR(45) NULL,`phoneNo` VARCHAR(45) NULL,`address` VARCHAR(45) NULL,`dob` DATE NULL,`joiningDate` DATE NULL,
`gender` VARCHAR(45) NULL, PRIMARY KEY (`doctorID`));
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE `patient`.`bill` (`billNo` INT NOT NULL,`admitID` INT NULL,`insuranceID` INT NULL,`totalAmount` INT NULL,`AmtPaid` INT NULL,`AmtPending` INT NULL,`paymentMode` VARCHAR(45) NULL,`Insured` TINY
INT NULL, PRIMARY KEY (`billNo`)) COMMENT = '     ';
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE `patient`.`bed` (`bedNo` INT NOT NULL,`wardID` INT NULL,`wardName` VARCHAR(45) NULL,`floor` INT NULL,`roomType` VARCHAR(45) NULL,`roomNo` INT NULL, PRIMARY KEY (`bedNo`));
Query OK, 0 rows affected (0.01 sec)

mysql> desc admission
    -> ;
```



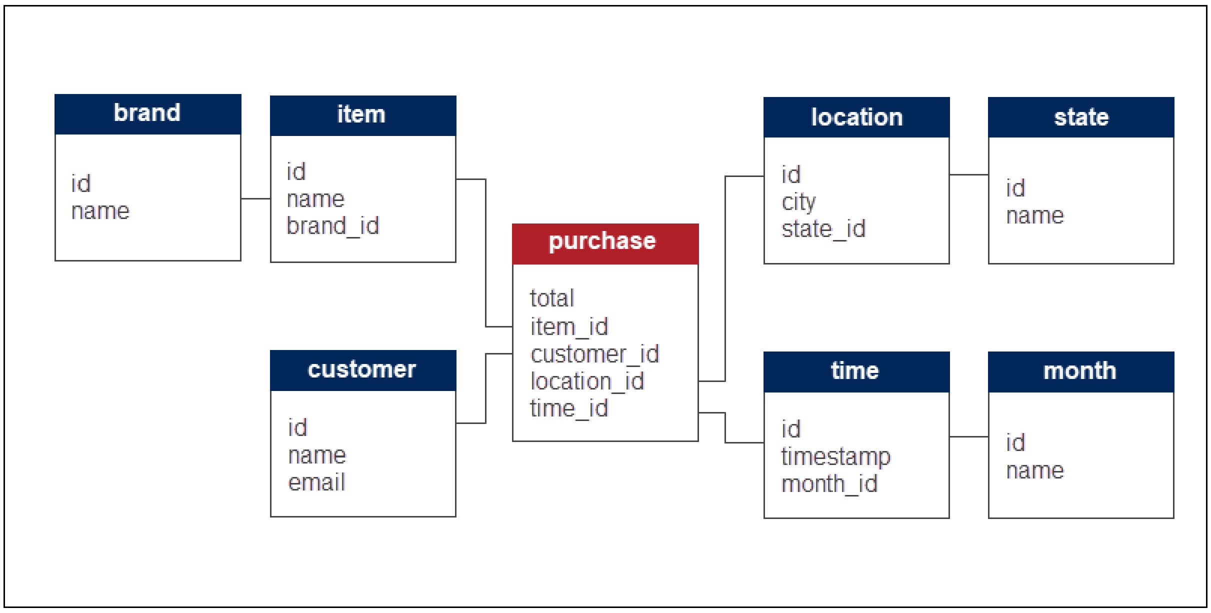
```
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql> a.create table admission (admitID int PRIMARY KEY, patientID int, doctorID int, billNo int, bedNo int, entryDate date, exitDate date)
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'a.create table admission (admitID int PRIMARY KEY, patie
ntID int, doctorID int, ' at line 1
mysql> cls
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'cls' at line 1
mysql> clear;
mysql> \system cls
--------------
mysql  Ver 8.1.0 for Win64 on x86_64 (MySQL Community Server - GPL)

Connection id:          14
Current database:
Current user:           root@localhost
SSL:                    Cipher in use is TLS_AES_256_GCM_SHA384
Using delimiter:        ;
Server version:         8.1.0 MySQL Community Server - GPL
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    utf8mb4
Db     characterset:    utf8mb4
Client characterset:    cp850
Conn.  characterset:    cp850
TCP port:               3306
Binary data as:         Hexadecimal
Uptime:                 9 min 40 sec

Threads: 2  Questions: 42  Slow queries: 0  Opens: 226  Flush tables: 3  Open tables: 142  Queries per second avg: 0.072
--------------

    -> ^C
mysql> cls
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'cls' at line 1
mysql> create table admission (admitID int PRIMARY KEY, patientID int, doctorID int, billNo int, bedNo int, entryDate date, exitDate date)
    -> ;
ERROR 1046 (3D000): No database selected
mysql> CLS
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'CLS' at line 1
mysql> ^C
```

Snowflake schema

- Snowflake schema is equivalent to the star schema. "A schema is known as a snowflake if one or more-dimension tables do not connect directly to the fact table but must join through other dimension tables."
- The snowflake schema is an expansion of the star schema where each point of the star explodes into more points. It is called snowflake schema because the diagram of snowflake schema resembles a snowflake. Snowflaking is a method of normalizing the dimension tables in a STAR schema. When we normalize all the dimension tables entirely, the resultant structure resembles a snowflake with the fact table in the middle.
- Snowflaking is used to develop the performance of specific queries. The schema is diagrammed with each fact surrounded by its associated dimensions, and those dimensions are related to other dimensions, branching out into a snowflake pattern.
- The snowflake schema consists of one fact table which is linked to many dimension tables, which can be linked to other dimension tables through a many-to-one relationship. Tables in a snowflake schema are generally normalized to the third normal form. Each dimension table performs exactly one level in a hierarchy.
- The following diagram shows a snowflake schema with two dimensions, each having three levels. A snowflake schema can have any number of dimensions, And each dimension can have any number of levels.



Example :



A snowflake schema is designed for flexible querying across more complex dimensions and relationship. It is suitable for many to many and one to many relationships between dimension levels.

Advantage of Snowflake Schema

1.The primary advantage of the snowflake schema is the development in query performance due to minimized disk storage requirements and joining smaller lookup tables.

2.It provides greater scalability in the interrelationship between dimension levels and components.

3.No redundancy, so it is easier to maintain.

Disadvantage of Snowflake Schema

1.The primary disadvantage of the snowflake schema is the additional maintenance efforts required due to the increasing number of lookup tables. It is also known as a multi fact star schema.

2.There are more complex queries and hence, difficult to understand.

3.More tables more join so more query execution time.

Implementation And Output:

Implementing a Snowflake schema in SQL involves organizing your data into a star schema with dimension tables and fact tables and then normalizing some of the dimension tables to further reduce redundancy. Here's a basic example of how to create a Snowflake schema using SQL:

Let's assume you are designing a data warehouse for a retail business, and you want to create a Snowflake schema for sales data.

1. Create Dimension Tables:
   Start by creating dimension tables for various attributes. These dimension tables contain information related to your business entities.

```
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(255),
    CategoryID INT
);
```

```
CREATE TABLE Category (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(255)
);
```

```
CREATE TABLE Store (
    StoreID INT PRIMARY KEY,
    StoreName VARCHAR(255),
    Location VARCHAR(255)
);
```

```
CREATE TABLE Date (
    DateID INT PRIMARY KEY,
    DateValue DATE,
    Day INT,
    Month INT,
    Year INT
);
```

2. Create Fact Table:
   Next, create a fact table that stores the sales transactions. This table will contain foreign keys to the dimension tables.

```
CREATE TABLE Sales (
    SalesID INT PRIMARY KEY,
    DateID INT,
    ProductID INT,
    StoreID INT,
    Quantity INT,
    Amount DECIMAL(10, 2)
);
```

3. Add Foreign Key Constraints:
   Define foreign key constraints to link the fact table to dimension tables.

```
ALTER TABLE Sales
ADD CONSTRAINT FK_Date FOREIGN KEY (DateID) REFERENCES
Date(DateID);

ALTER TABLE Sales
ADD CONSTRAINT FK_Product FOREIGN KEY (ProductID) REFERENCES
Product(ProductID);

ALTER TABLE Sales
ADD CONSTRAINT FK_Store FOREIGN KEY (StoreID) REFERENCES
Store(StoreID);
```

4. Populate the Tables:
   Insert data into your dimension and fact tables based on your business data.

5. Additional Normalization (Optional):
   In a Snowflake schema, you can further normalize dimension tables to reduce redundancy. For example, you can split the 'Category' table into separate 'Category' and 'SubCategory' tables if it makes sense for your data.

```
CREATE TABLE SubCategory (
    SubCategoryID INT PRIMARY KEY,
    SubCategoryName VARCHAR(255),
    CategoryID INT
);
```

   Then, update the 'Product' table to reference the 'SubCategory' table instead of the 'Category' table.

```
ALTER TABLE Product
ADD CONSTRAINT FK_SubCategory FOREIGN KEY (CategoryID)
REFERENCES SubCategory(SubCategoryID);
```

This is a basic example of how to implement a Snowflake schema in SQL. Depending on your specific business requirements and data, you may need to create more tables and relationships. Remember that Snowflake schemas are designed to normalize data and reduce redundancy, which can improve query performance and maintainability in data warehousing scenarios.

Sales (Fact Table)

- SalesID (Primary Key)
- DateID (Foreign Key to Date)
- ProductID (Foreign Key to Product)

Date (Dimension Table)

- DateID (Primary Key)
- DateValue

Product (Dimension Table)

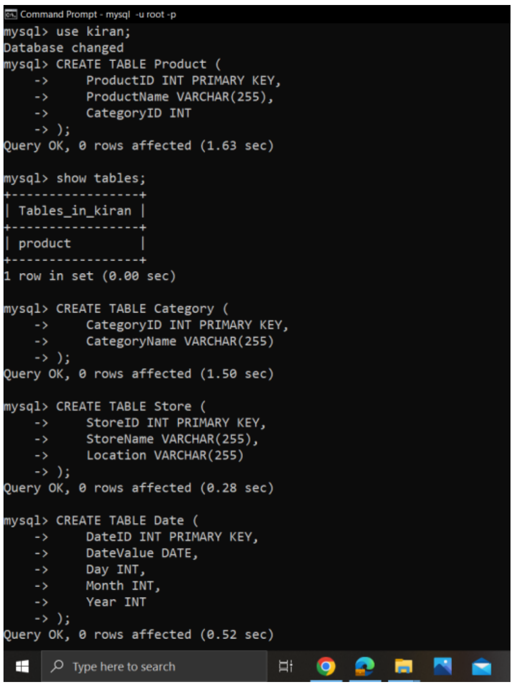
- ProductID (Primary Key)
- ProductName
- CategoryID (Foreign Key to Category)

Category (Dimension Table)

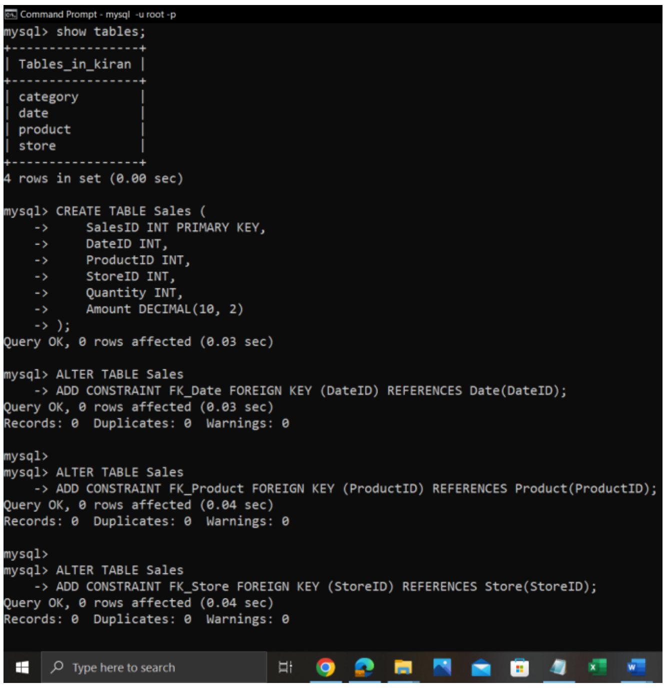
- CategoryID (Primary Key)
- CategoryName

Store (Dimension Table)

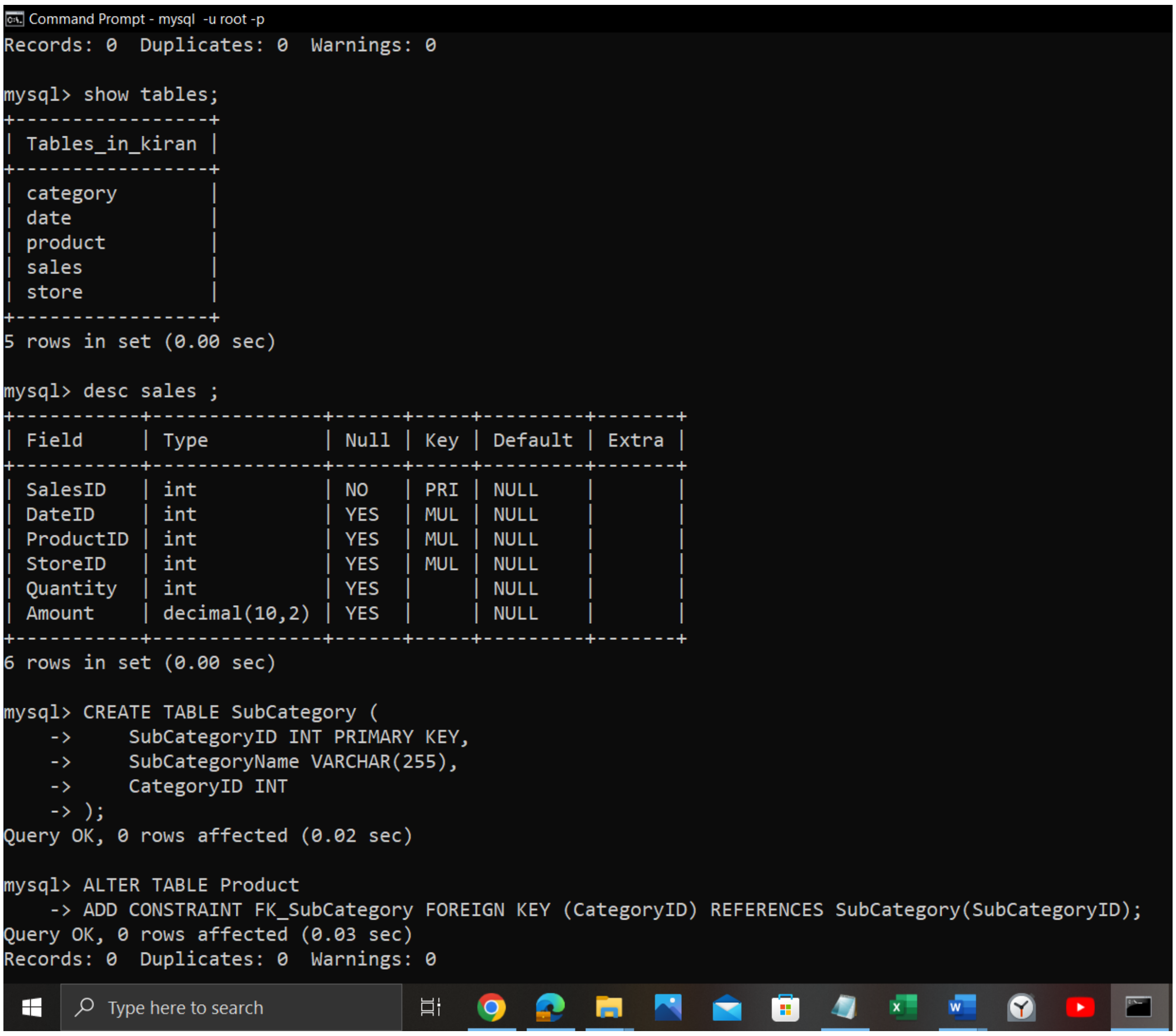
- StoreID (Primary Key)
- StoreName



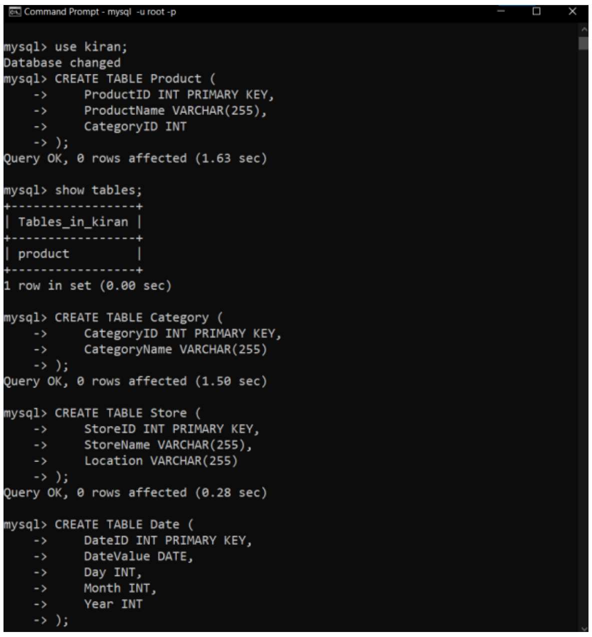
```
Command Prompt - mysql -u root -p
mysql> use kiran;
Database changed
mysql> CREATE TABLE Product (
    ->     ProductID INT PRIMARY KEY,
    ->     ProductName VARCHAR(255),
    ->     CategoryID INT
    -> );
Query OK, 0 rows affected (1.63 sec)

mysql> show tables;
+----------------+
| Tables_in_kiran |
+----------------+
| product        |
+----------------+
1 row in set (0.00 sec)

mysql> CREATE TABLE Category (
    ->     CategoryID INT PRIMARY KEY,
    ->     CategoryName VARCHAR(255)
    -> );
Query OK, 0 rows affected (1.50 sec)

mysql> CREATE TABLE Store (
    ->     StoreID INT PRIMARY KEY,
    ->     StoreName VARCHAR(255),
    ->     Location VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.28 sec)

mysql> CREATE TABLE Date (
    ->     DateID INT PRIMARY KEY,
    ->     DateValue DATE,
    ->     Day INT,
    ->     Month INT,
    ->     Year INT
    -> );
Query OK, 0 rows affected (0.52 sec)
```



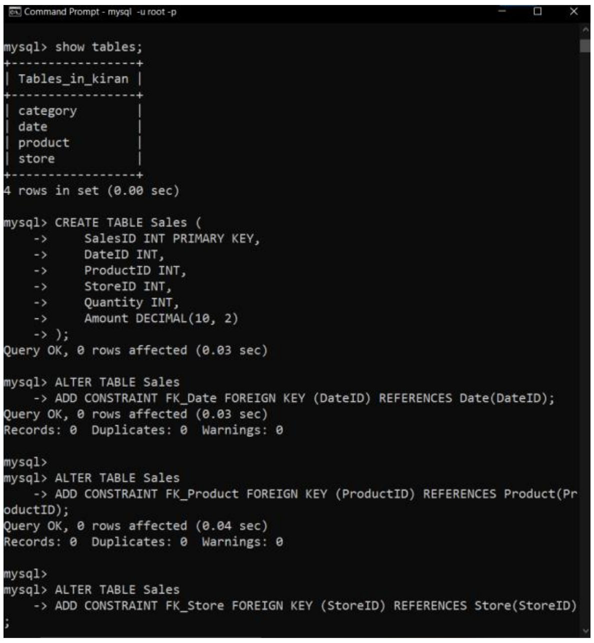
```
Command Prompt - mysql -u root -p
mysql> show tables;
+----------------+
| Tables_in_kiran |
+----------------+
| category       |
| date           |
| product        |
| store          |
+----------------+
4 rows in set (0.00 sec)

mysql> CREATE TABLE Sales (
    ->     SalesID INT PRIMARY KEY,
    ->     DateID INT,
    ->     ProductID INT,
    ->     StoreID INT,
    ->     Quantity INT,
    ->     Amount DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT FK_Date FOREIGN KEY (DateID) REFERENCES Date(DateID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT FK_Product FOREIGN KEY (ProductID) REFERENCES Product(ProductID);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT FK_Store FOREIGN KEY (StoreID) REFERENCES Store(StoreID);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



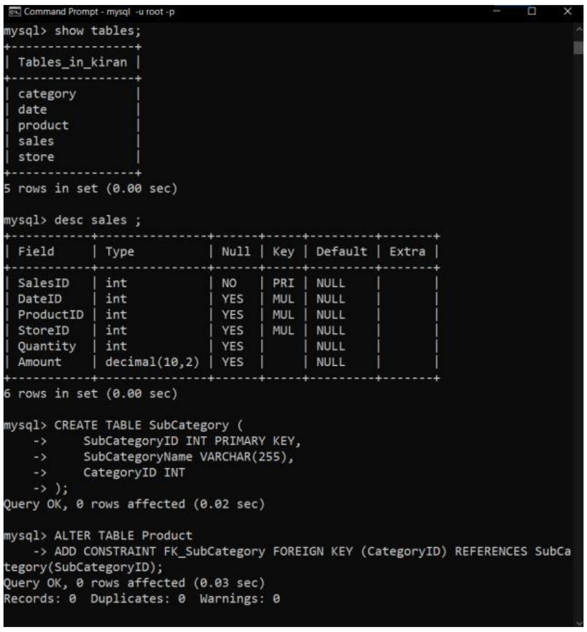
```
Command Prompt - mysql -u root -p
Records: 0  Duplicates: 0  Warnings: 0

mysql> show tables;
+-----------------+
| Tables_in_kiran |
+-----------------+
| category        |
| date            |
| product         |
| sales           |
| store           |
+-----------------+
5 rows in set (0.00 sec)

mysql> desc sales ;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| SalesID   | int           | NO   | PRI | NULL    |       |
| DateID    | int           | YES  | MUL | NULL    |       |
| ProductID | int           | YES  | MUL | NULL    |       |
| StoreID   | int           | YES  | MUL | NULL    |       |
| Quantity  | int           | YES  |     | NULL    |       |
| Amount    | decimal(10,2) | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> CREATE TABLE SubCategory (
    ->     SubCategoryID INT PRIMARY KEY,
    ->     SubCategoryName VARCHAR(255),
    ->     CategoryID INT
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE Product
    -> ADD CONSTRAINT FK_SubCategory FOREIGN KEY (CategoryID) REFERENCES SubCategory(SubCategoryID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



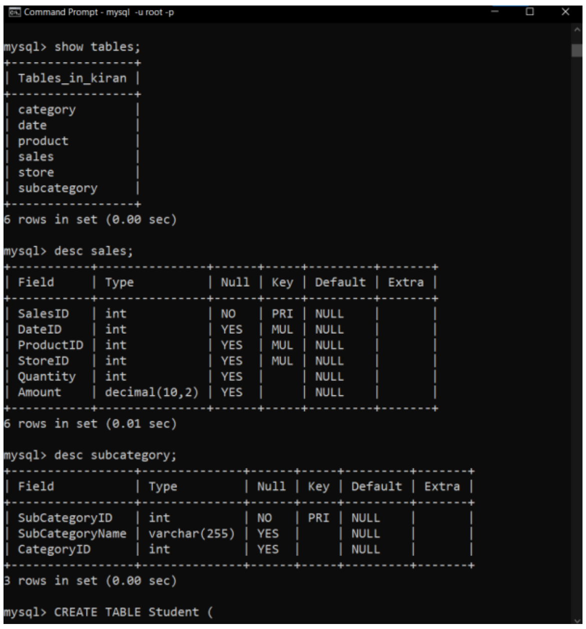
```
mysql> use kiran;
Database changed
mysql> CREATE TABLE Product (
    ->     ProductID INT PRIMARY KEY,
    ->     ProductName VARCHAR(255),
    ->     CategoryID INT
    -> );
Query OK, 0 rows affected (1.63 sec)

mysql> show tables;
+-----------------+
| Tables_in_kiran |
+-----------------+
| product         |
+-----------------+
1 row in set (0.00 sec)

mysql> CREATE TABLE Category (
    ->     CategoryID INT PRIMARY KEY,
    ->     CategoryName VARCHAR(255)
    -> );
Query OK, 0 rows affected (1.50 sec)

mysql> CREATE TABLE Store (
    ->     StoreID INT PRIMARY KEY,
    ->     StoreName VARCHAR(255),
    ->     Location VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.28 sec)

mysql> CREATE TABLE Date (
    ->     DateID INT PRIMARY KEY,
    ->     DateValue DATE,
    ->     Day INT,
    ->     Month INT,
    ->     Year INT
    -> );
```



```
mysql> show tables;
+-----------------+
| Tables_in_kiran |
+-----------------+
| category        |
| date            |
| product         |
| store           |
+-----------------+
4 rows in set (0.00 sec)

mysql> CREATE TABLE Sales (
    ->     SalesID INT PRIMARY KEY,
    ->     DateID INT,
    ->     ProductID INT,
    ->     StoreID INT,
    ->     Quantity INT,
    ->     Amount DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT FK_Date FOREIGN KEY (DateID) REFERENCES Date(DateID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT FK_Product FOREIGN KEY (ProductID) REFERENCES Product(Pr
oductID);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT FK_Store FOREIGN KEY (StoreID) REFERENCES Store(StoreID)
;
```



```
mysql> show tables;
+-----------------+
| Tables_in_kiran |
+-----------------+
| category        |
| date            |
| product         |
| sales           |
| store           |
+-----------------+
5 rows in set (0.00 sec)

mysql> desc sales ;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| SalesID   | int           | NO   | PRI | NULL    |       |
| DateID    | int           | YES  | MUL | NULL    |       |
| ProductID | int           | YES  | MUL | NULL    |       |
| StoreID   | int           | YES  | MUL | NULL    |       |
| Quantity  | int           | YES  |     | NULL    |       |
| Amount    | decimal(10,2) | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> CREATE TABLE SubCategory (
    ->     SubCategoryID INT PRIMARY KEY,
    ->     SubCategoryName VARCHAR(255),
    ->     CategoryID INT
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE Product
    -> ADD CONSTRAINT FK_SubCategory FOREIGN KEY (CategoryID) REFERENCES SubCa
tegory(SubCategoryID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



```
mysql> show tables;
+-----------------+
| Tables_in_kiran |
+-----------------+
| category        |
| date            |
| product         |
| sales           |
| store           |
| subcategory     |
+-----------------+
6 rows in set (0.00 sec)

mysql> desc sales;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| SalesID   | int           | NO   | PRI | NULL    |       |
| DateID    | int           | YES  | MUL | NULL    |       |
| ProductID | int           | YES  | MUL | NULL    |       |
| StoreID   | int           | YES  | MUL | NULL    |       |
| Quantity  | int           | YES  |     | NULL    |       |
| Amount    | decimal(10,2) | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> desc subcategory;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| SubCategoryID   | int          | NO   | PRI | NULL    |       |
| SubCategoryName | varchar(255) | YES  |     | NULL    |       |
| CategoryID      | int          | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> CREATE TABLE Student (
```

Fact Constellation Schema

A Fact Constellation Schema, also known as a galaxy schema, is a data warehouse design that includes multiple fact tables sharing dimension tables. Each fact table in the constellation represents a different business process or set of measures. Here's a basic example of how to implement a Fact Constellation Schema in SQL:

In this example, we'll design a data warehouse for a university, focusing on student enrollment and course registration data.

1. Create Dimension Tables:
   Start by creating dimension tables for various attributes shared across multiple fact tables.

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    FirstName VARCHAR(255),
    LastName VARCHAR(255),
    Birthdate DATE,
    MajorID INT
);

CREATE TABLE Major (
    MajorID INT PRIMARY KEY,
    MajorName VARCHAR(255)
);

CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    CourseName VARCHAR(255),
    DepartmentID INT
);

CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(255)
);
```

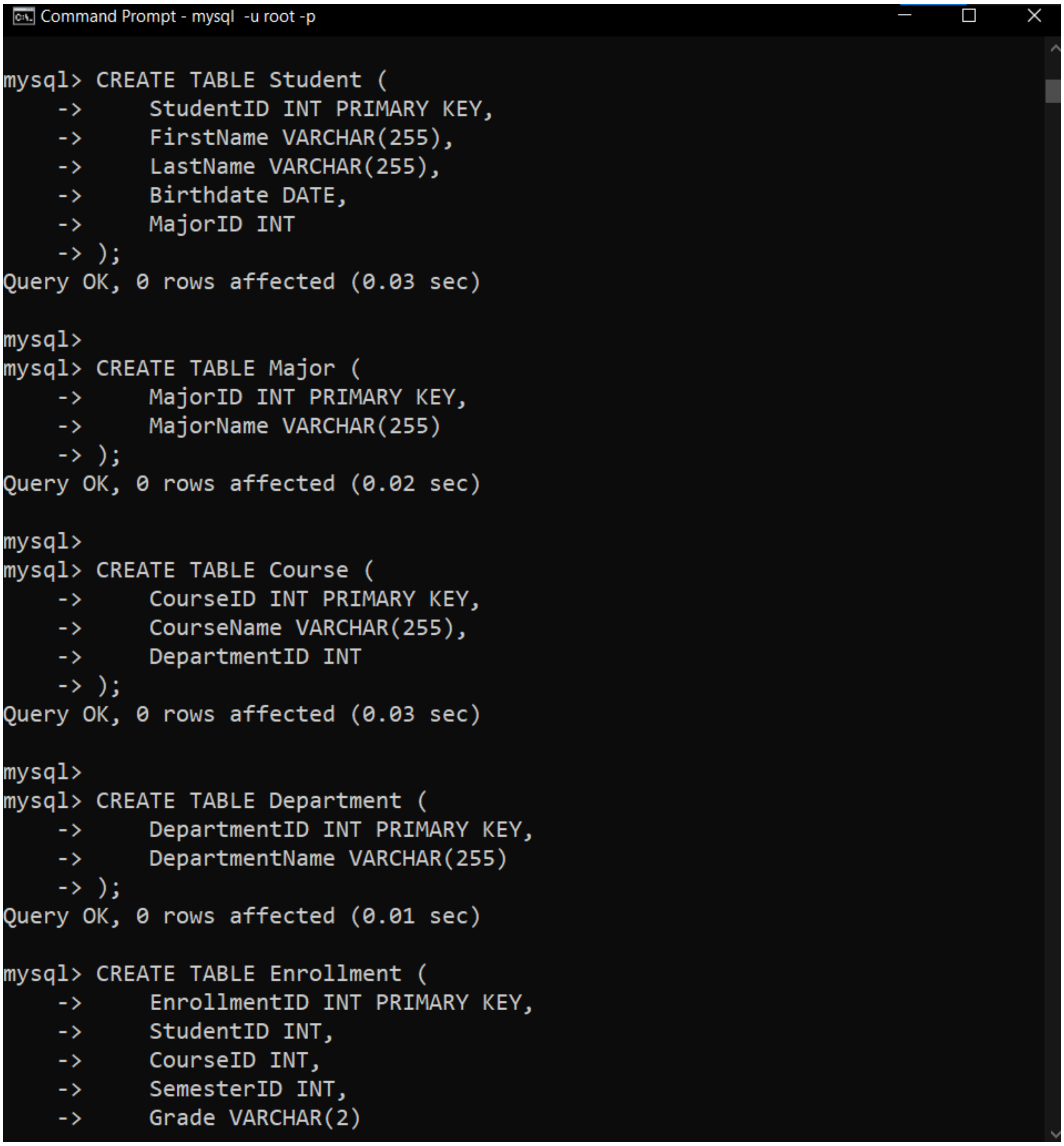
2. Add Foreign Key Constraints:
   Define foreign key constraints to link the fact tables to dimension tables.

```
ALTER TABLE Enrollment
ADD CONSTRAINT FK_Student_Enrollment FOREIGN KEY (StudentID) REFERENCES Student(StudentID);
```

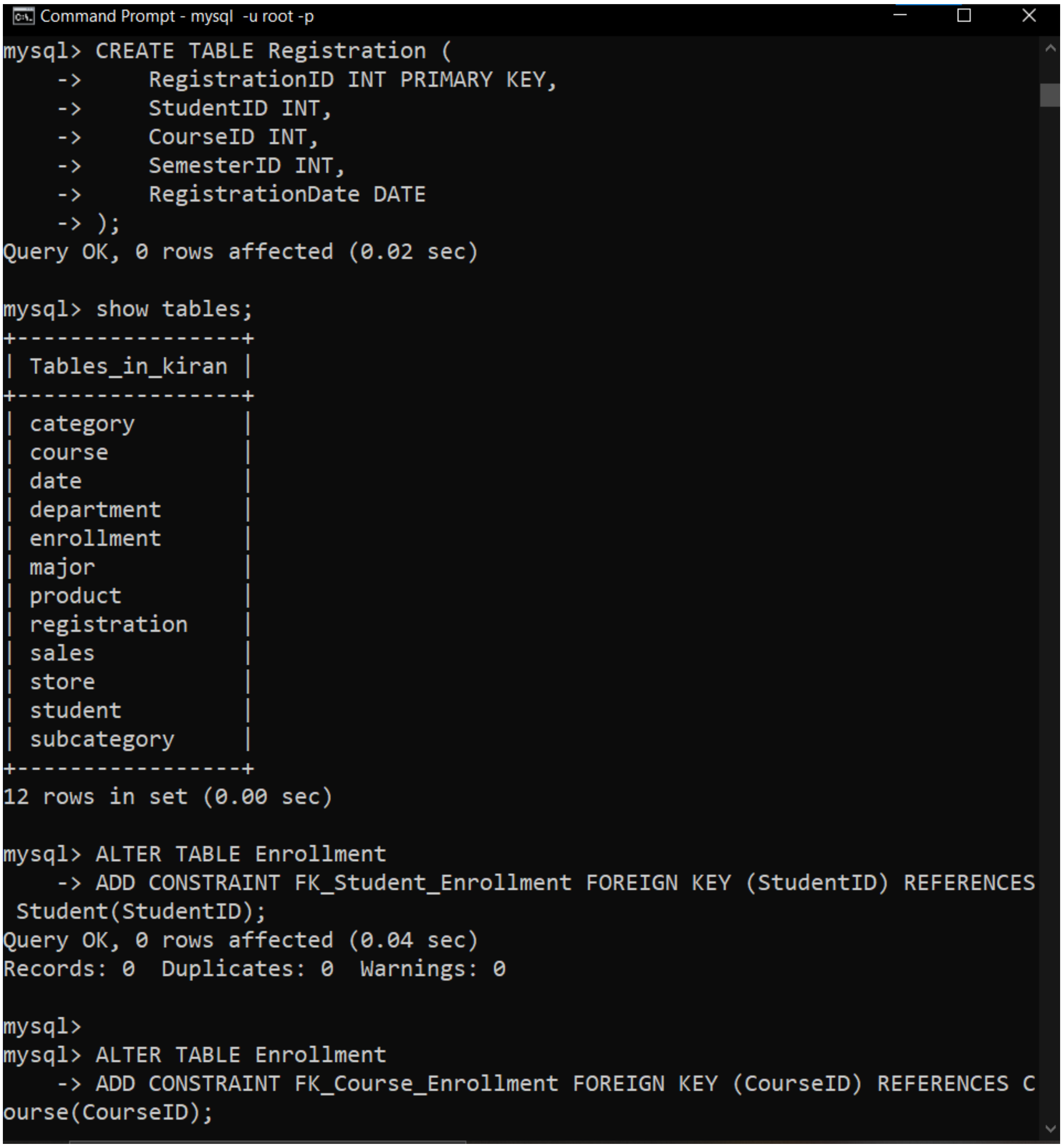
```
ALTER TABLE Enrollment
ADD CONSTRAINT FK_Course_Enrollment FOREIGN KEY (CourseID) REFERENCES Course(CourseID);

ALTER TABLE Registration
ADD CONSTRAINT FK_Student_Registration FOREIGN KEY (StudentID) REFERENCES Student(StudentID);

ALTER TABLE Registration
ADD CONSTRAINT FK_Course_Registration FOREIGN KEY (CourseID) REFERENCES Course(CourseID);
```

3. Populate the Tables:
   Insert data into your dimension and fact tables based on your university's student and course data.

In this Fact Constellation Schema example, we have two fact tables (Enrollment and Registration) sharing common dimension tables (Student, Course, Major, Department, Semester). Each fact table represents a different aspect of the university's data, allowing you to analyze student enrollment and course registration separately while leveraging the same dimensions for consistency.

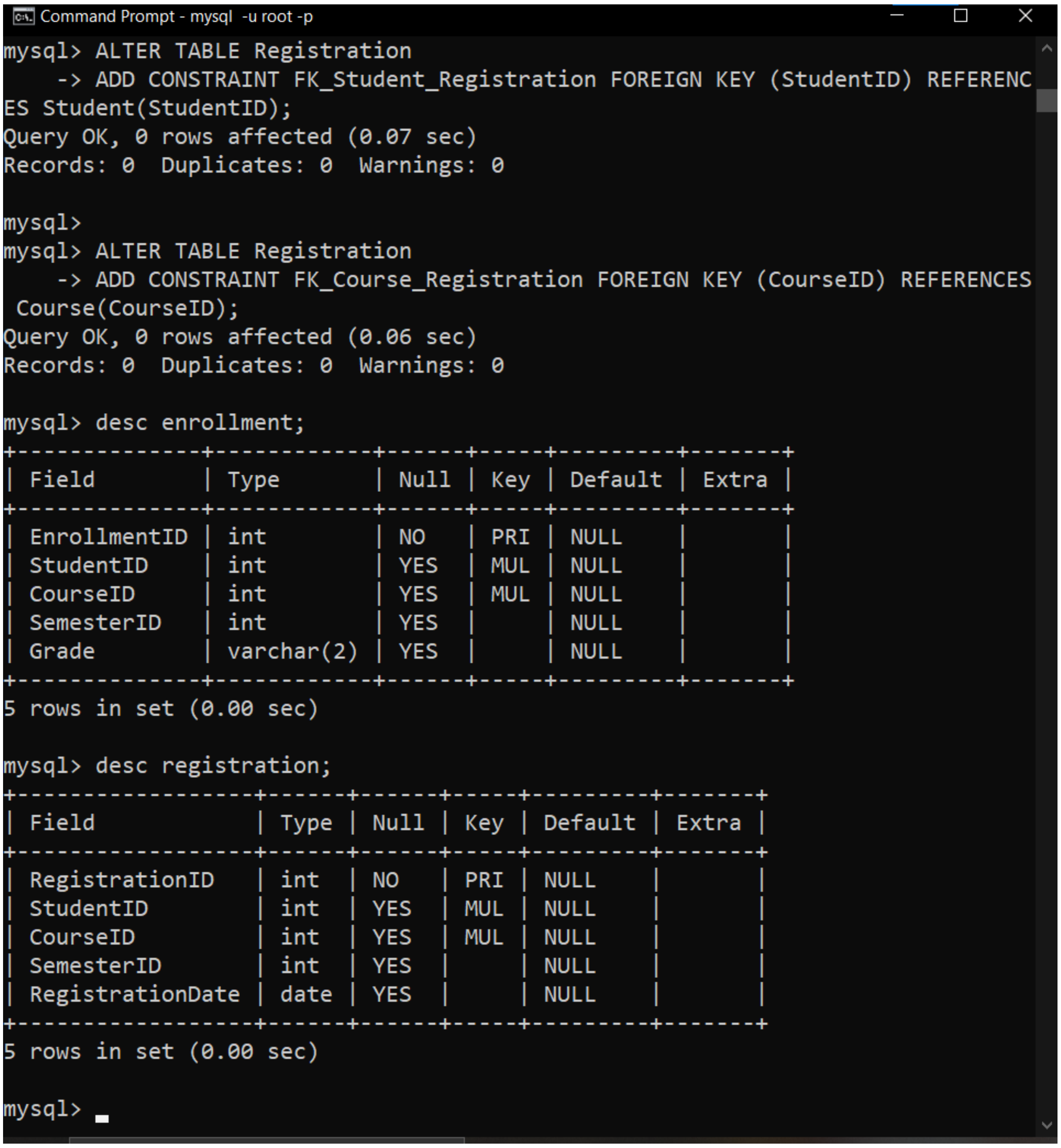


```
mysql> CREATE TABLE Student (
    ->     StudentID INT PRIMARY KEY,
    ->     FirstName VARCHAR(255),
    ->     LastName VARCHAR(255),
    ->     Birthdate DATE,
    ->     MajorID INT
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> CREATE TABLE Major (
    ->     MajorID INT PRIMARY KEY,
    ->     MajorName VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE Course (
    ->     CourseID INT PRIMARY KEY,
    ->     CourseName VARCHAR(255),
    ->     DepartmentID INT
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> CREATE TABLE Department (
    ->     DepartmentID INT PRIMARY KEY,
    ->     DepartmentName VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Enrollment (
    ->     EnrollmentID INT PRIMARY KEY,
    ->     StudentID INT,
    ->     CourseID INT,
    ->     SemesterID INT,
    ->     Grade VARCHAR(2)
```



Command Prompt - mysql -u root -p

```
mysql> CREATE TABLE Registration (
    ->     RegistrationID INT PRIMARY KEY,
    ->     StudentID INT,
    ->     CourseID INT,
    ->     SemesterID INT,
    ->     RegistrationDate DATE
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> show tables;
+------------------+
| Tables_in_kiran |
+------------------+
| category         |
| course           |
| date             |
| department       |
| enrollment       |
| major            |
| product          |
| registration     |
| sales            |
| store            |
| student          |
| subcategory      |
+------------------+
12 rows in set (0.00 sec)

mysql> ALTER TABLE Enrollment
    -> ADD CONSTRAINT FK_Student_Enrollment FOREIGN KEY (StudentID) REFERENCES
 Student(StudentID);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Enrollment
    -> ADD CONSTRAINT FK_Course_Enrollment FOREIGN KEY (CourseID) REFERENCES C
ourse(CourseID);
```


Command Prompt - mysql -u root -p

```
mysql> ALTER TABLE Registration
    -> ADD CONSTRAINT FK_Student_Registration FOREIGN KEY (StudentID) REFERENC
ES Student(StudentID);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Registration
    -> ADD CONSTRAINT FK_Course_Registration FOREIGN KEY (CourseID) REFERENCES
 Course(CourseID);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc enrollment;
+--------------+------------+------+-----+---------+-------+
| Field        | Type       | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| EnrollmentID | int        | NO   | PRI | NULL    |       |
| StudentID    | int        | YES  | MUL | NULL    |       |
| CourseID     | int        | YES  | MUL | NULL    |       |
| SemesterID   | int        | YES  |     | NULL    |       |
| Grade        | varchar(2) | YES  |     | NULL    |       |
+--------------+------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> desc registration;
+------------------+------+------+-----+---------+-------+
| Field            | Type | Null | Key | Default | Extra |
+------------------+------+------+-----+---------+-------+
| RegistrationID   | int  | NO   | PRI | NULL    |       |
| StudentID        | int  | YES  | MUL | NULL    |       |
| CourseID         | int  | YES  | MUL | NULL    |       |
| SemesterID       | int  | YES  |     | NULL    |       |
| RegistrationDate | date | YES  |     | NULL    |       |
+------------------+------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql>
```

Conclusion: In conclusion, database schema design plays a crucial role in organizing and optimizing data warehousing solutions. The Star Schema offers simplicity and efficient query performance but may involve data redundancy. Snowflake Schema normalizes data to reduce redundancy while maintaining data integrity. Fact Constellation Schema, with its multiple fact tables, provides flexibility for complex scenarios but requires more intricate maintenance. The choice among these schema types should align with specific business needs, balancing performance, data integrity, and complexity for an effective data warehousing solution.