ADBMS Assignment No. 5

Q.1.) Figure 9.8 shows an ER schema for a database that can be used to keep track of transport ships and their locations for maritime authorities. Map this schema into a relational schema and specify all primary keys and foreign keys.

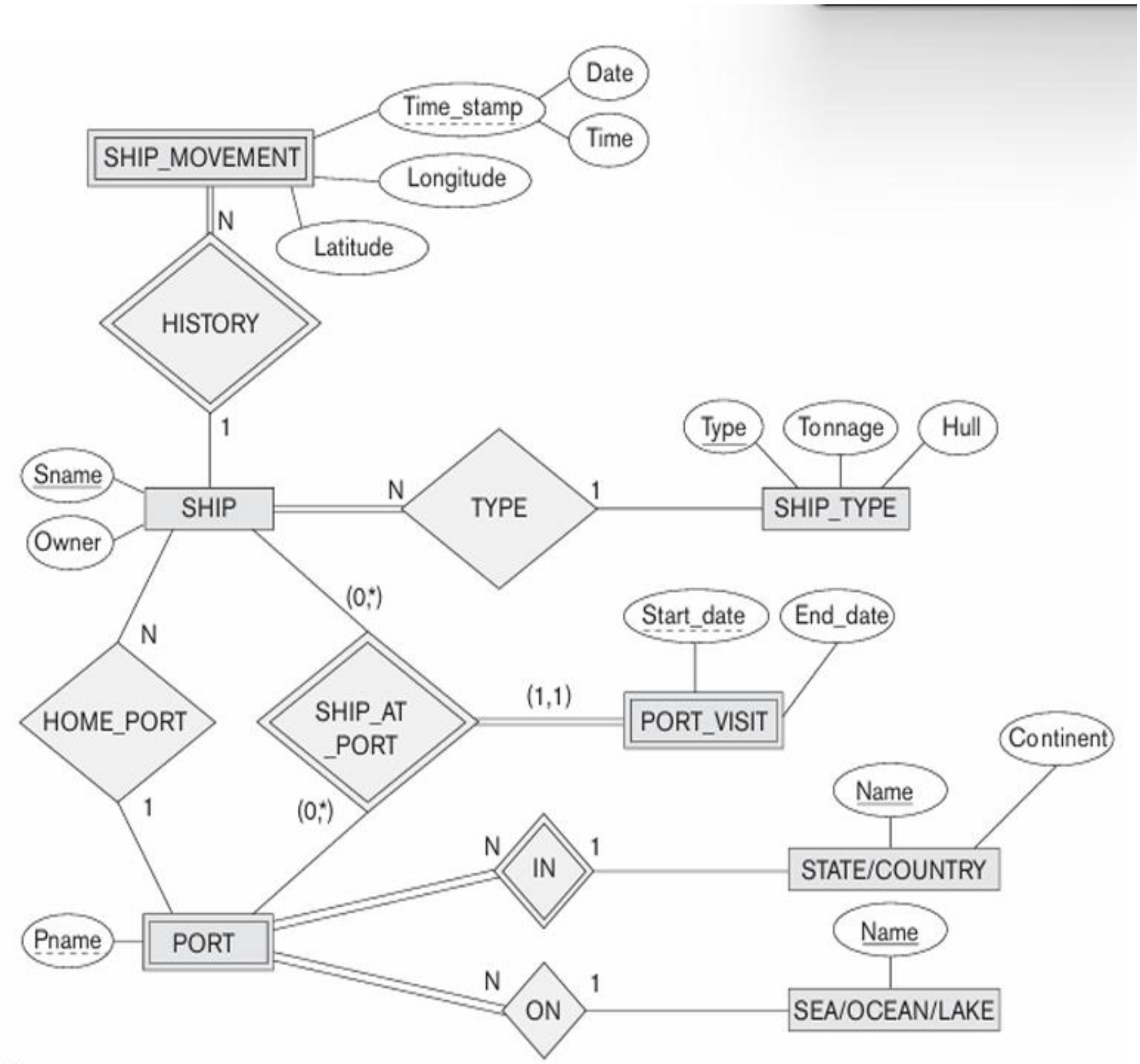


Figure 9.8
An ER schema for a SHIP_TRACKING database.

Ans :-

Here's the relational schema mapped from the ER diagram in Figure 9.8, including **primary keys (PK)** and **foreign keys (FK)**:

1. SHIP

```sql
SHIP(Sname PRIMARY KEY, Owner, Type)
```

- `Type` is a **foreign key** referencing `SHIP_TYPE(Type)`

2. SHIP_TYPE

```sql
SHIP_TYPE(Type PRIMARY KEY, Tonnage, Hull)
```

3. HOME_PORT

This is a relationship between `SHIP` and `PORT`:

```sql
HOME_PORT(Sname PRIMARY KEY, Pname)
```

- `Sname` is a **foreign key** referencing `SHIP(Sname)`
- `Pname` is a **foreign key** referencing `PORT(Pname)`

4. PORT

```sql
PORT(Pname PRIMARY KEY)
```

5. SHIP_AT_PORT

This relationship links `SHIP` and `PORT`, with an associative entity `PORT_VISIT`:

```sql
SHIP_AT_PORT(Sname, Pname, Start_date PRIMARY KEY, End_date)
```

- **Composite PK**: `(Sname, Pname, Start_date)`
- `Sname` is a **foreign key** referencing `SHIP(Sname)`
- `Pname` is a **foreign key** referencing `PORT(Pname)`

6. SHIP_MOVEMENT

```sql
SHIP_MOVEMENT(Sname, Time_stamp PRIMARY KEY, Latitude, Longitude, Date, Time)
```

- **Composite PK**: `(Sname, Time_stamp)`
- `Sname` is a **foreign key** referencing `SHIP(Sname)`

7. STATE_COUNTRY

```sql
STATE_COUNTRY(Name PRIMARY KEY, Continent)
```

8. SEA_OCEAN_LAKE

```sql
SEA_OCEAN_LAKE(Name PRIMARY KEY)
```

9. IN

This relationship links `PORT` to `STATE_COUNTRY`:

```sql
IN(Pname PRIMARY KEY, Country_Name)
```

- `Pname` is a **foreign key** referencing `PORT(Pname)`
- `Country_Name` is a **foreign key** referencing `STATE_COUNTRY(Name)`

10. ON

This relationship links `PORT` to `SEA_OCEAN_LAKE`:

```sql
ON(Pname PRIMARY KEY, Water_Body_Name)
```

- `Pname` is a **foreign key** referencing `PORT(Pname)`
- `Water_Body_Name` is a **foreign key** referencing `SEA_OCEAN_LAKE(Name)`

Q.2.) Consider the EER diagram in Figure 9.9 for a car dealer. Map the EER schema into a set of relations. For the VEHICLE to CAR/TRUCK/SUV generalization, consider the four options presented in Section 9.2.1and show the relational schema design under each of those options.

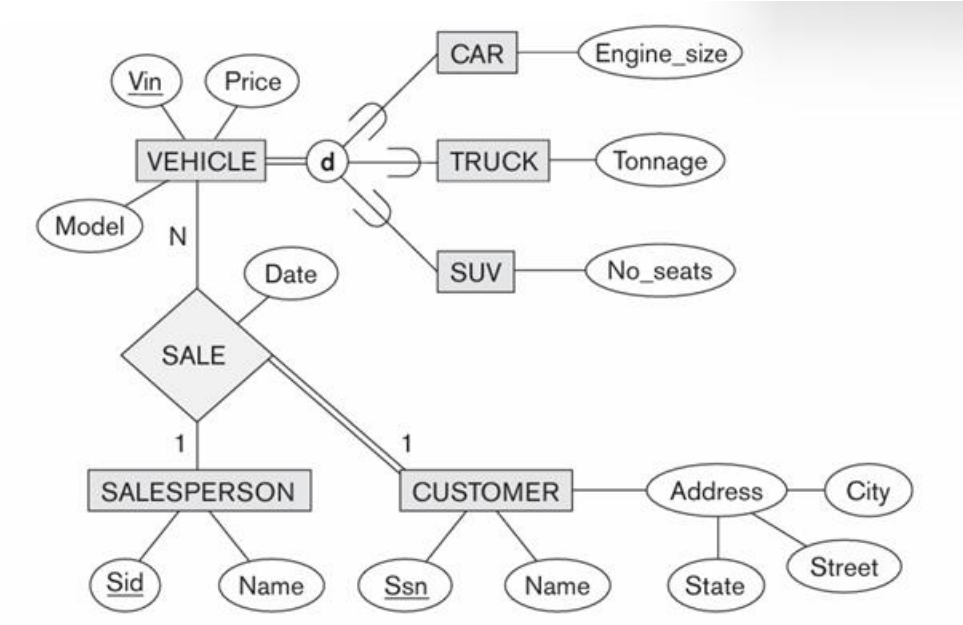


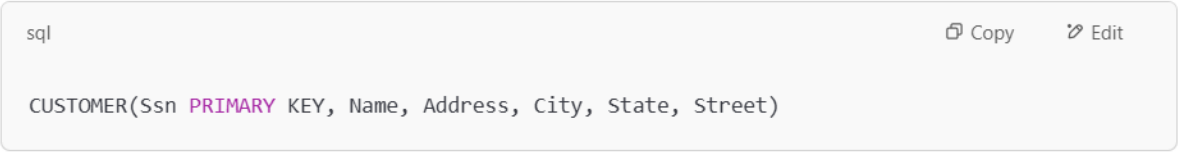
Figure 9.9
EER diagram for a car dealer

Ans :- Let's walk through the mapping of the **EER diagram (Figure 9.9)** for a **car dealer** to relational schemas, focusing especially on the **specialization/generalization** of VEHICLE into CAR, TRUCK, and SUV.

Entities and Relationships (Baseline)

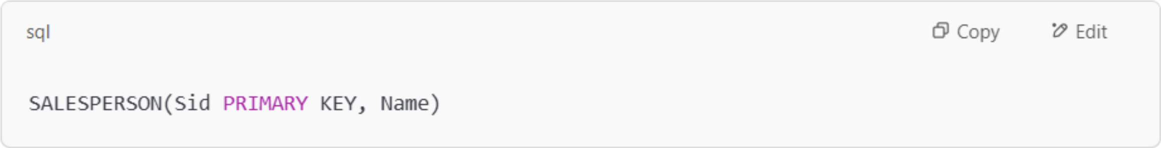
Before tackling the generalization options, we convert the rest of the EER diagram into relations:

1. CUSTOMER



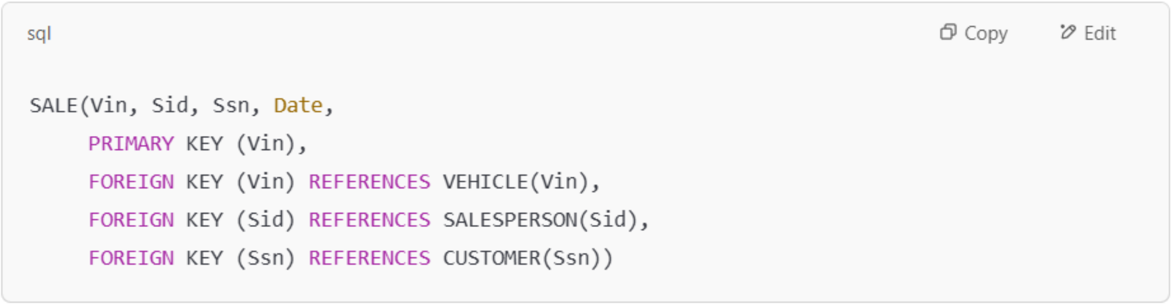
```sql
CUSTOMER(Ssn PRIMARY KEY, Name, Address, City, State, Street)
```

2. SALESPERSON



```sql
SALESPERSON(Sid PRIMARY KEY, Name)
```

3. SALE



```sql
SALE(Vin, Sid, Ssn, Date,
     PRIMARY KEY (Vin),
     FOREIGN KEY (Vin) REFERENCES VEHICLE(Vin),
     FOREIGN KEY (Sid) REFERENCES SALESPERSON(Sid),
     FOREIGN KEY (Ssn) REFERENCES CUSTOMER(Ssn))
```

Now, let’s focus on the **VEHICLE generalization** using the **four mapping options** from **Section 9.2.1**:

Option 1: Multiple Relations - Superclass and Subclass Relations

- One relation for the superclass (VEHICLE)
- One relation for each subclass

```
VEHICLE(Vin PRIMARY KEY, Model, Price)

CAR(Vin PRIMARY KEY, Engine_size, FOREIGN KEY (Vin) REFERENCES VEHICLE(Vin))

TRUCK(Vin PRIMARY KEY, Tonnage, FOREIGN KEY (Vin) REFERENCES VEHICLE(Vin))

SUV(Vin PRIMARY KEY, No_seats, FOREIGN KEY (Vin) REFERENCES VEHICLE(Vin))
```

  - Pros: Easy to query general vehicle info
  - Cons: Requires joins to get subclass-specific attributes

Option 2: Multiple Relations - Subclass Relations Only

- No superclass table
- Each subclass has its own full set of attributes (repeats superclass attributes)

```
CAR(Vin PRIMARY KEY, Model, Price, Engine_size)

TRUCK(Vin PRIMARY KEY, Model, Price, Tonnage)

SUV(Vin PRIMARY KEY, Model, Price, No_seats)
```

- Pros: No joins needed
- Cons: Redundancy and potential inconsistencies

Option 3: Single Relation with NULLs

- All classes merged into a single relation with nullable fields

```
VEHICLE(
  Vin PRIMARY KEY,
  Model,
  Price,
  Engine_size,    -- for CAR
  Tonnage,        -- for TRUCK
  No_seats        -- for SUV
)
```

- Pros: One unified table
- Cons: Many nulls unless all attributes are shared

Option 4: Single Relation with Discriminator Column

- One unified relation with a type discriminator

```
VEHICLE(
  Vin PRIMARY KEY,
  Model,
  Price,
  Vehicle_Type CHECK (Vehicle_Type IN ('CAR', 'TRUCK', 'SUV')),
  Engine_size,
  Tonnage,
  No_seats
)
```

  - Pros: Centralized; easy filtering using Vehicle_Type
  - Cons: Similar null problem; logic needed to enforce valid combinations

Q.3.) Describe the features of the above university EER diagram i.e entities, their relations, cardinality, superclass , subclass relations , union, disjoint, overlapping constructs , min , max constraints if any. Map the same EER schema into relational schema.

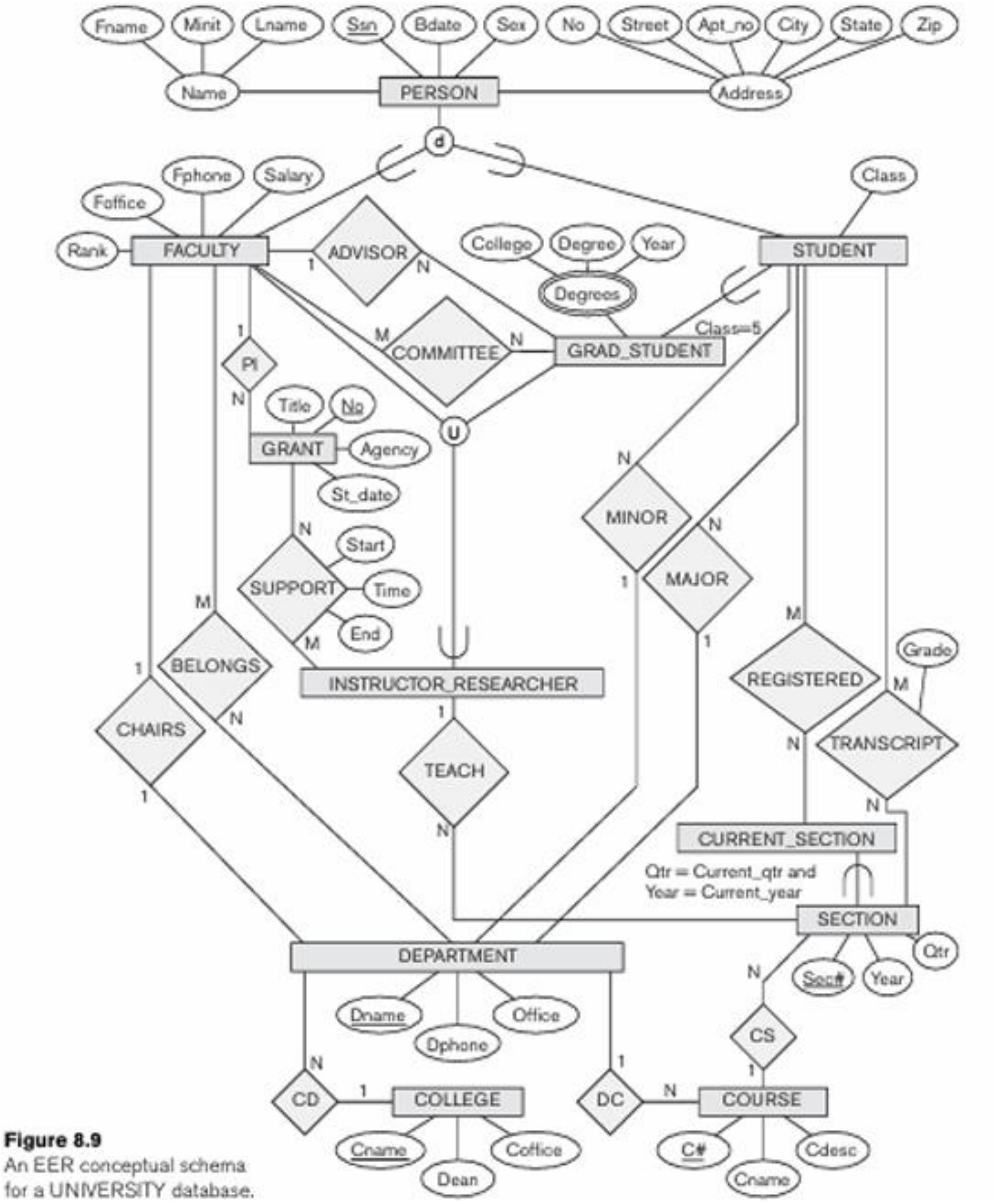


Figure 8.9
An EER conceptual schema for a UNIVERSITY database.

Ans:- **Description of the University EER Diagram (Figure 8.9)**

The EER diagram represents a **University database** with detailed features including entities, relationships, specialization/generalization, union types, and constraints.

◆ Entities and Attributes:

1. PERSON

- **Attributes**: `Ssn (PK)`, `Fname`, `Minit`, `Lname`, `Bdate`, `Sex`, `Address (Street, Apt_no, City, State, Zip)`

2. FACULTY (Subclass of PERSON)

- **Attributes**: `Foffice`, `Fphone`, `Salary`, `Rank`

3. STUDENT (Subclass of PERSON)

- **Attributes**: `Class`

4. GRAD_STUDENT (Subclass of STUDENT)

- **Attributes**: `College`, `Degree`, `Year`

5. INSTRUCTOR_RESEARCHER (Union Type of FACULTY and GRAD_STUDENT)

- No direct attributes

6. DEPARTMENT

- **Attributes**: `Dname (PK)`, `Office`, `Dphone`

7. COLLEGE

- **Attributes**: `Cname (PK)`, `Coffice`, `Dean`

8. COURSE

- **Attributes**: `C# (PK)`, `Cname`, `Cdesc`

9. SECTION

- **Attributes**: `Sect#`, `Year`, `Qtr` (PK is combination: `C#`, `Sect#`, `Year`, `Qtr`)

10. CURRENT_SECTION

- Subset of SECTION (for current term)

11. GRANT

- **Attributes**: `Title`, `No (PK)`, `Agency`, `St_date`

12. SUPPORT

- **Attributes**: `Start`, `End`, `Time`

🔸 Relationships & Their Features:

Relationship	Participating Entities	Cardinality	Notes
ADVISOR	STUDENT – FACULTY	N:1	A student has one advisor
COMMITTEE	GRAD_STUDENT – FACULTY	N:N	Advisory committees
PI	FACULTY – GRANT	1:N	PI (Principal Investigator) for grants
SUPPORT	GRAD_STUDENT – GRANT	N:N	With attributes
BELONGS	DEPARTMENT – COLLEGE	M:1	Each department belongs to one college
CHAIRS	DEPARTMENT – FACULTY	1:1	Chair of department
TEACH	INSTRUCTOR_RESEARCHER – SECTION	N:N	Teaches sections
REGISTERED	STUDENT – SECTION	M:N	With `Grade` attribute (TRANSCRIPT)
MAJOR	STUDENT – DEPARTMENT	N:1	Each student majors in 1 department
MINOR	STUDENT – DEPARTMENT	N:N	Students can have multiple minors

◆ Specialization/Generalization/Union:

Construct Type	Entities Involved	Notes
Disjoint Specialization	PERSON → FACULTY, STUDENT	d notation (disjoint)
Overlapping Specialization	STUDENT → GRAD_STUDENT	o notation (overlapping)
Union Type	INSTRUCTOR_RESEARCHER ← FACULTY ∪ GRAD_STUDENT	Denoted by U symbol

◆ Constraints:

- **Min–Max Cardinalities**:
  - Many relationships use `(1, N)`, `(M, 1)`, etc., shown on connecting lines
- **Subset Entity**:
  - CURRENT_SECTION is a subset of SECTION (conditions: current quarter and year)

🧩 Relational Schema Mapping

PERSON

sql Copy Edit

```sql
PERSON(Ssn PRIMARY KEY, Fname, Minit, Lname, Bdate, Sex, Street, Apt_no, City, State, Zip)
```

FACULTY

sql Copy Edit

```sql
FACULTY(Ssn PRIMARY KEY, Foffice, Fphone, Salary, Rank,
        FOREIGN KEY(Ssn) REFERENCES PERSON(Ssn))
```

STUDENT

```sql
STUDENT(Ssn PRIMARY KEY, Class,
        FOREIGN KEY(Ssn) REFERENCES PERSON(Ssn))
```

GRAD_STUDENT

```sql
GRAD_STUDENT(Ssn PRIMARY KEY, College, Degree, Year,
             FOREIGN KEY(Ssn) REFERENCES STUDENT(Ssn))
```

INSTRUCTOR_RESEARCHER

```sql
INSTRUCTOR_RESEARCHER(Ssn PRIMARY KEY,
                      FOREIGN KEY(Ssn) REFERENCES FACULTY(Ssn),
                      FOREIGN KEY(Ssn) REFERENCES GRAD_STUDENT(Ssn))
```

DEPARTMENT

```sql
DEPARTMENT(Dname PRIMARY KEY, Office, Dphone)
```

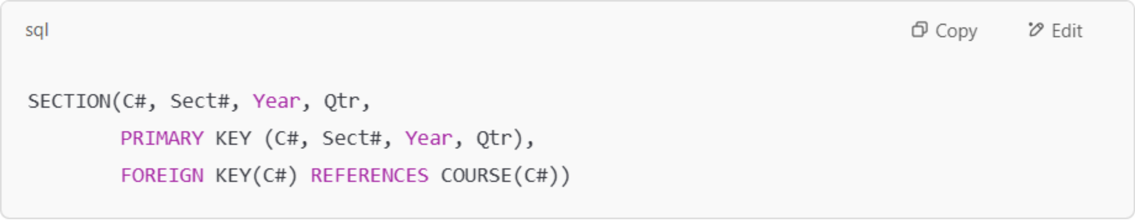
COLLEGE

```sql
COLLEGE(Cname PRIMARY KEY, Coffice, Dean)
```

COURSE

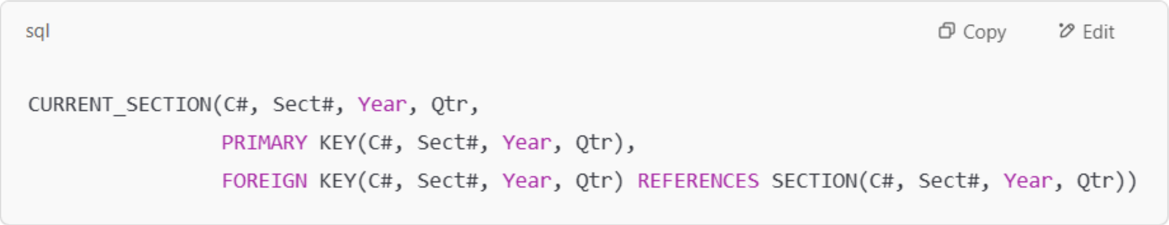
```sql
COURSE(C# PRIMARY KEY, Cname, Cdesc)
```

SECTION



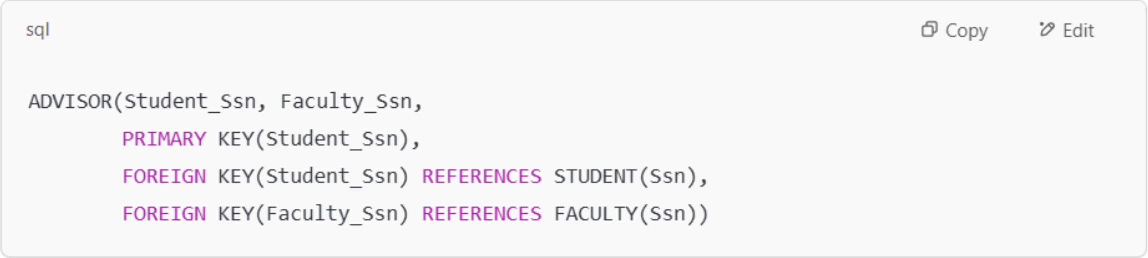
```
SECTION(C#, Sect#, Year, Qtr,
        PRIMARY KEY (C#, Sect#, Year, Qtr),
        FOREIGN KEY(C#) REFERENCES COURSE(C#))
```

CURRENT_SECTION



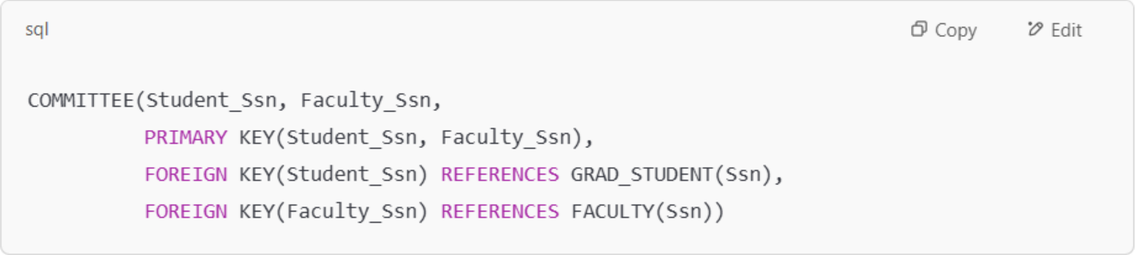
```
CURRENT_SECTION(C#, Sect#, Year, Qtr,
                PRIMARY KEY(C#, Sect#, Year, Qtr),
                FOREIGN KEY(C#, Sect#, Year, Qtr) REFERENCES SECTION(C#, Sect#, Year, Qtr))
```

ADVISOR



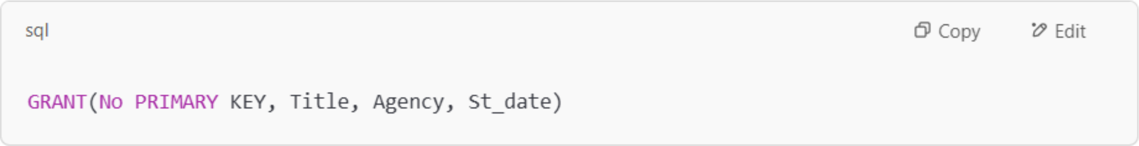
```
ADVISOR(Student_Ssn, Faculty_Ssn,
        PRIMARY KEY(Student_Ssn),
        FOREIGN KEY(Student_Ssn) REFERENCES STUDENT(Ssn),
        FOREIGN KEY(Faculty_Ssn) REFERENCES FACULTY(Ssn))
```

COMMITTEE



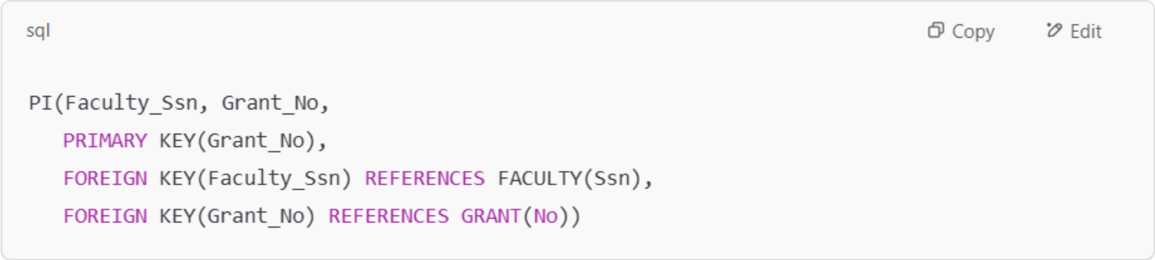
```
COMMITTEE(Student_Ssn, Faculty_Ssn,
          PRIMARY KEY(Student_Ssn, Faculty_Ssn),
          FOREIGN KEY(Student_Ssn) REFERENCES GRAD_STUDENT(Ssn),
          FOREIGN KEY(Faculty_Ssn) REFERENCES FACULTY(Ssn))
```

GRANT



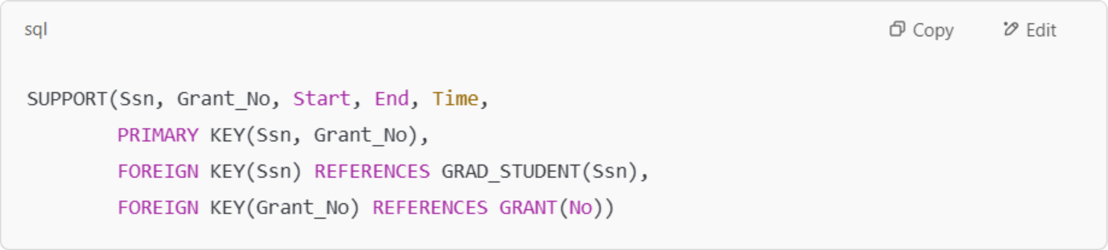
```
GRANT(No PRIMARY KEY, Title, Agency, St_date)
```

PI



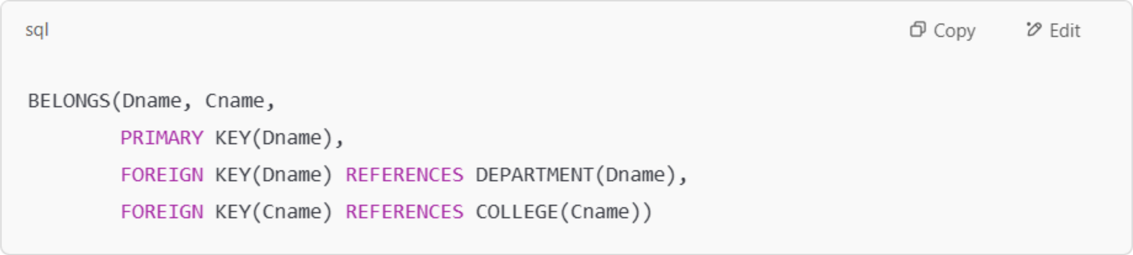
```sql
PI(Faculty_Ssn, Grant_No,
   PRIMARY KEY(Grant_No),
   FOREIGN KEY(Faculty_Ssn) REFERENCES FACULTY(Ssn),
   FOREIGN KEY(Grant_No) REFERENCES GRANT(No))
```

SUPPORT



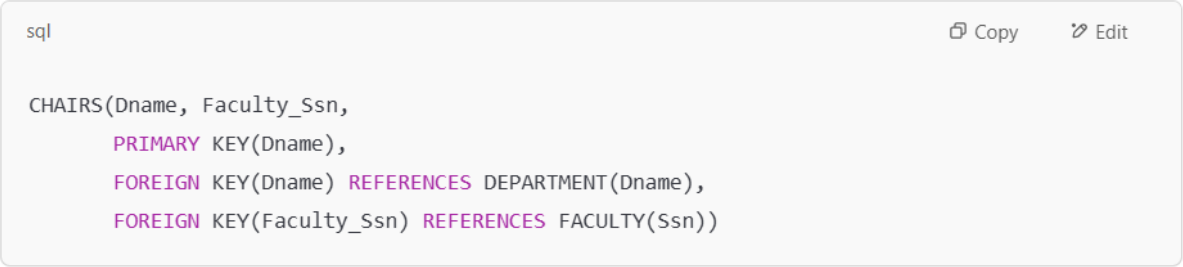
```sql
SUPPORT(Ssn, Grant_No, Start, End, Time,
        PRIMARY KEY(Ssn, Grant_No),
        FOREIGN KEY(Ssn) REFERENCES GRAD_STUDENT(Ssn),
        FOREIGN KEY(Grant_No) REFERENCES GRANT(No))
```

BELONGS



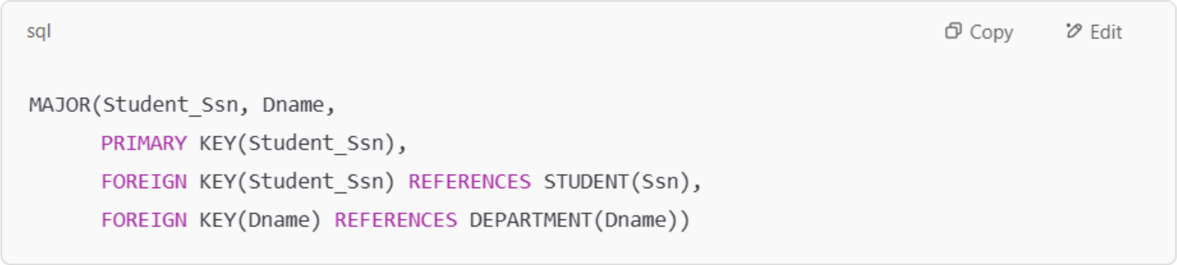
```sql
BELONGS(Dname, Cname,
        PRIMARY KEY(Dname),
        FOREIGN KEY(Dname) REFERENCES DEPARTMENT(Dname),
        FOREIGN KEY(Cname) REFERENCES COLLEGE(Cname))
```

CHAIRS



```sql
CHAIRS(Dname, Faculty_Ssn,
       PRIMARY KEY(Dname),
       FOREIGN KEY(Dname) REFERENCES DEPARTMENT(Dname),
       FOREIGN KEY(Faculty_Ssn) REFERENCES FACULTY(Ssn))
```

MAJOR



```sql
MAJOR(Student_Ssn, Dname,
      PRIMARY KEY(Student_Ssn),
      FOREIGN KEY(Student_Ssn) REFERENCES STUDENT(Ssn),
      FOREIGN KEY(Dname) REFERENCES DEPARTMENT(Dname))
```

MINOR

sql Copy Edit

```sql
MINOR(Student_Ssn, Dname,
      PRIMARY KEY(Student_Ssn, Dname),
      FOREIGN KEY(Student_Ssn) REFERENCES STUDENT(Ssn),
      FOREIGN KEY(Dname) REFERENCES DEPARTMENT(Dname))
```

REGISTERED / TRANSCRIPT

sql Copy Edit

```sql
TRANSCRIPT(Ssn, C#, Sect#, Year, Qtr, Grade,
           PRIMARY KEY(Ssn, C#, Sect#, Year, Qtr),
           FOREIGN KEY(Ssn) REFERENCES STUDENT(Ssn),
           FOREIGN KEY(C#, Sect#, Year, Qtr) REFERENCES SECTION(C#, Sect#, Year, Qtr))
```