Babysitter Management System

Name	ID	Contribution
Shah Sanzida Masiat	20-42937-1	Use case diagram, Class diagram , Er diagram , Schema Diagram, Scenario description ,
Md. Apon Riaz Talukdar	20-42905-1	User Interface, Table creation , Data Insert , query Writing ,PL/SQL
Argho Proshad Singho	20-42906-1	Introduction, Activity diagram Project Proposal, Conclusion
Md. Borhan Uddin	20-44002-2	Normalization, Relational Algebra

Course : Advance Database Management System

Section : A

Session : Summer 22-23

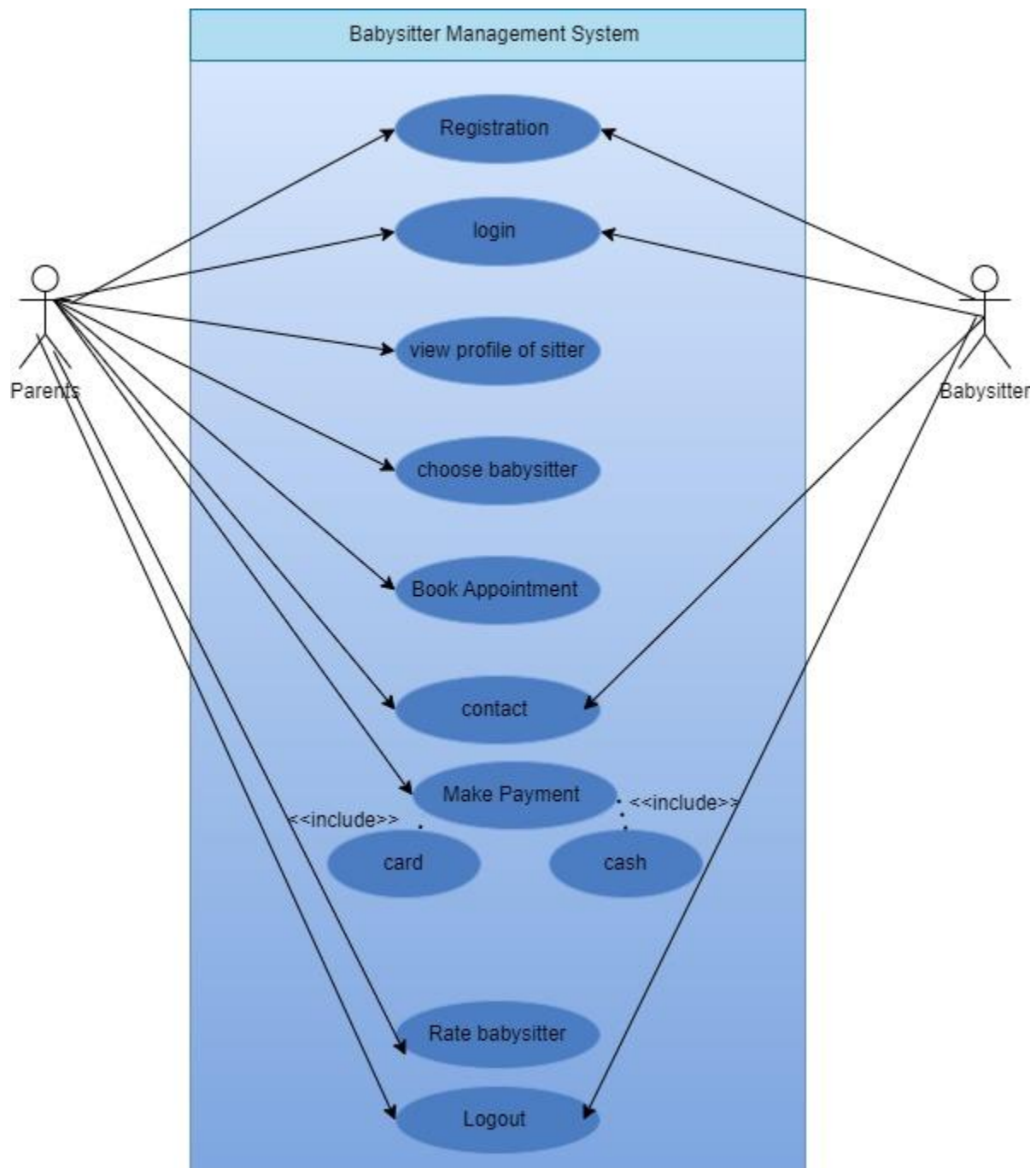Faculty: Juena Ahmed Noshin

Contents

Introduction:

The Babysitter Management System is a revolutionary technology that is changing the way modern families and childcare companies handle their babysitting requirements. Our solution is intended to simplify and improve the process of easily discovering, scheduling, and managing babysitters. With our Babysitter Management System, you can say goodbye to the hassles of last-minute babysitter searches and stressful scheduling disputes. Parents can quickly search through a pool of reputable and qualified babysitters, examine their profiles, availability, and ratings, and effortlessly plan appointments that meet their needs using our revolutionary platform. Meanwhile, babysitters may take use of an easy-to-use interface to manage their schedules, contact with parents, and enhance their reputation in the childcare community. The Babysitter Management System is set to change the way parents and babysitters communicate and interact, by offering for busy families, a dependable and effective daycare option. Join us as we begin on this exciting adventure together to simplify babysitter management!"
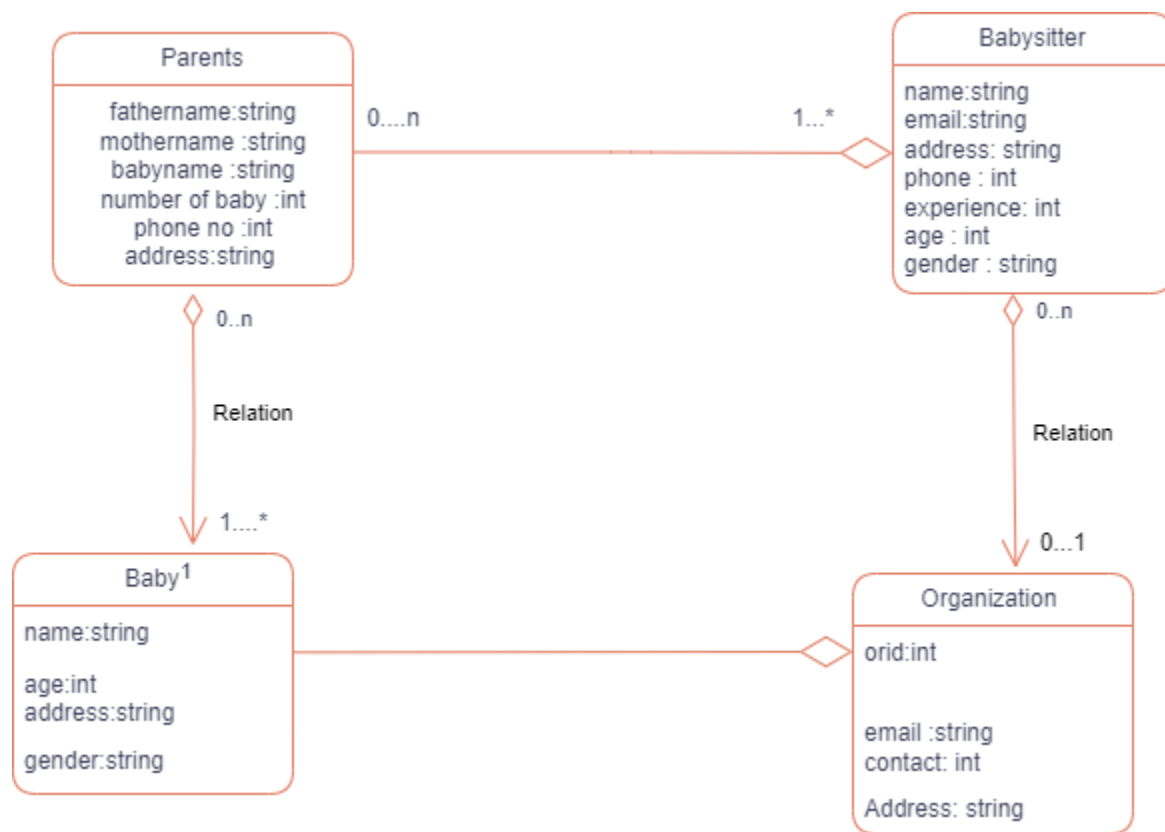
Project Proposal:

The objective of this organization is to handle the requirement of babysitting for the new modern families . We aim to promote this revolutionary technology to the parents' day to day life and make their life a lot more easier. Join us as we embark on this exciting journey to simplify babysitter management. User Registration and Login: Parents and babysitters will create individual accounts to access the system. Babysitter Profile Management: Babysitters can create and update their profiles, including information about their skills, experience, availability, and rates. Parent Interface: Parents can search for babysitters based on various criteria such as location, availability, and ratings. They can view detailed profiles, reviews, and ratings before making a selection. Appointment Scheduling: Parents can schedule appointments with selected babysitters based on their availability and specific requirements. Communication Platform: The system will provide a secure messaging platform for parents and babysitters to communicate and discuss appointment details. Review and Rating System: Parents can leave reviews and ratings for babysitters based on their experience, which helps build a reliable reputation system. Notifications and Reminders: The system will send automated notifications and reminders to parents and babysitters regarding upcoming appointments.

Use case diagram :

Babysitter Management System

Registration

login

view profile of sitter

choose babysitter

Book Appointment

contact

Make Payment

<<include>>

card

<<include>>

cash

Rate babysitter

Logout

Parents

Babysitter

Class diagram :

Parents

fathername:string
mothername :string
babyname :string
number of baby :int
phone no :int
address:string

0....n ———————————— 1...*

Babysitter

name:string
email:string
address: string
phone : int
experience: int
age : int
gender : string

0..n

Relation

0..n

Relation

1....*

Baby[1]

name:string

age:int
address:string

gender:string

0...1

Organization

orid:int

email :string
contact: int

Address: string

Activity Diagram :

User Interface
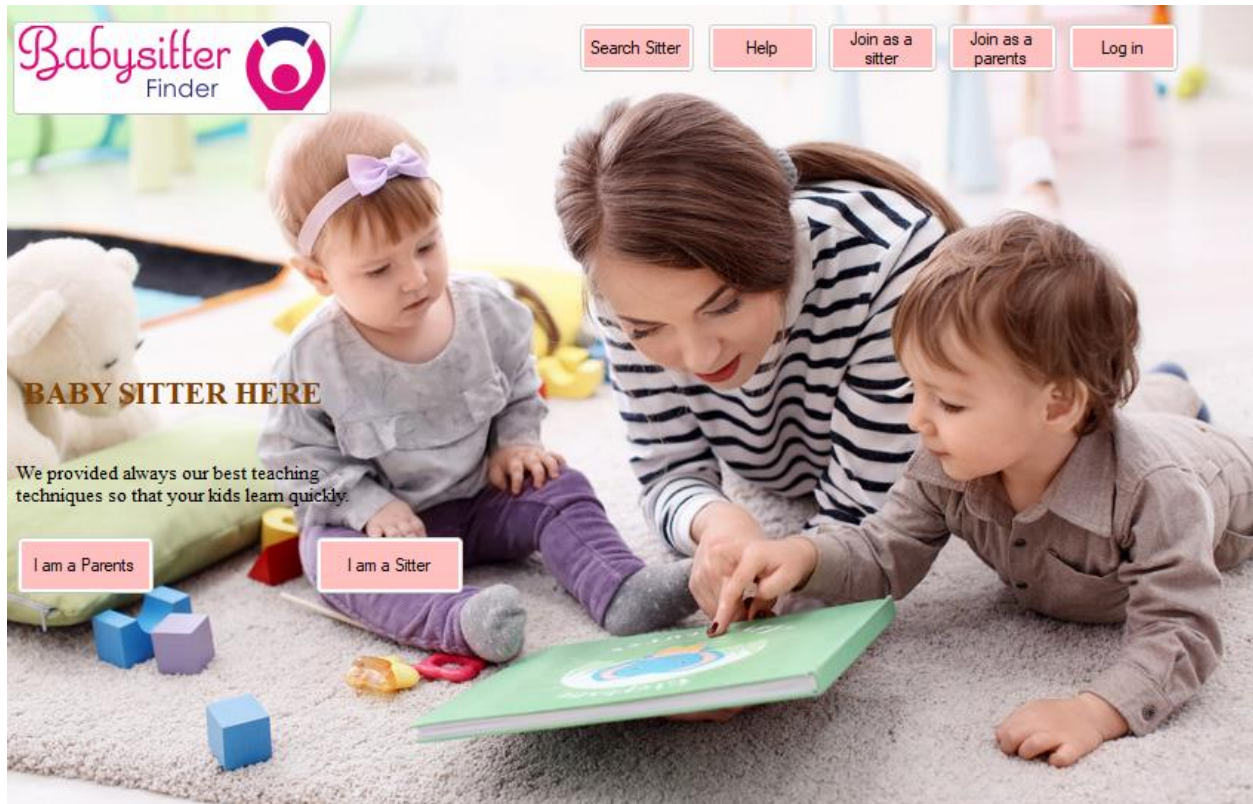
Home Page

Parents Login

Babysitter Page
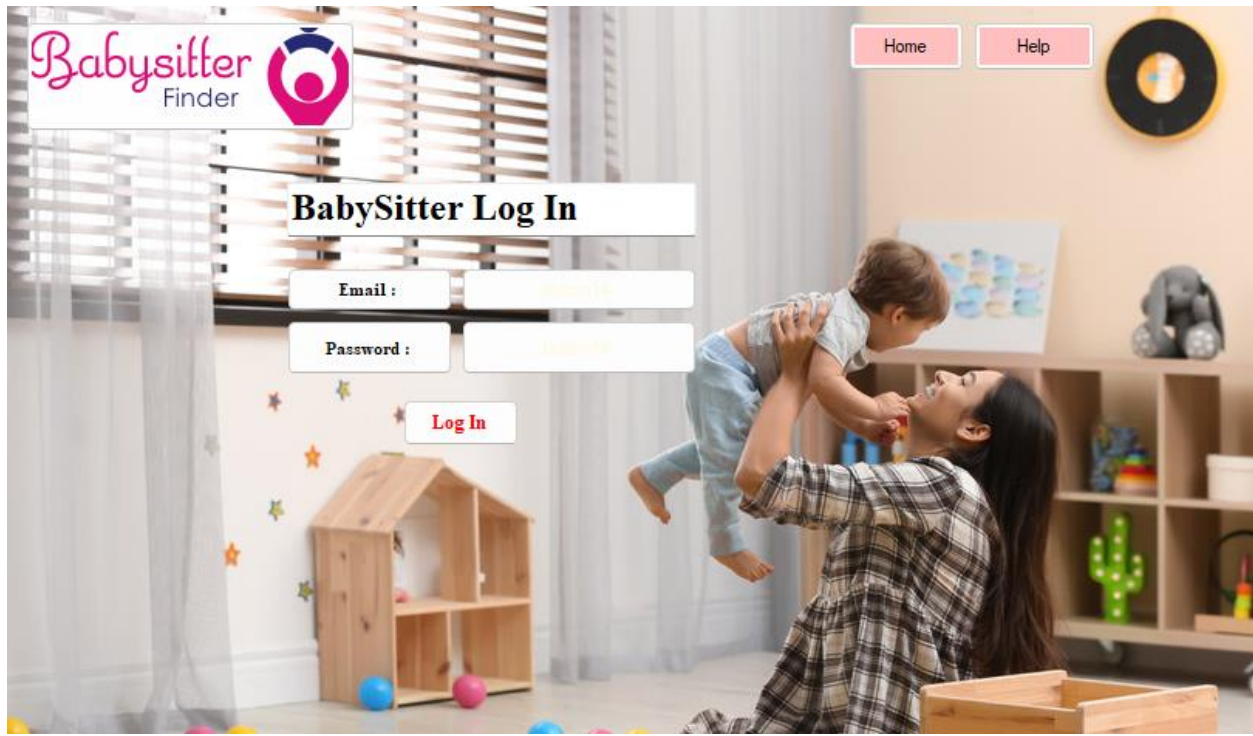
Babysitter Login

Searching Babysitter

Project Scenario Description :

 In the Babysitter Management System parents and babysitters connect and manage childcare needs. The parent has to register into the system for receiving its offerings. Parents interested in utilizing the Babysitter Management System must first register. During the registration process, parents provide essential information that uniquely identifies them within the system. This information includes a ParentId, a distinct identifier assigned to each parent.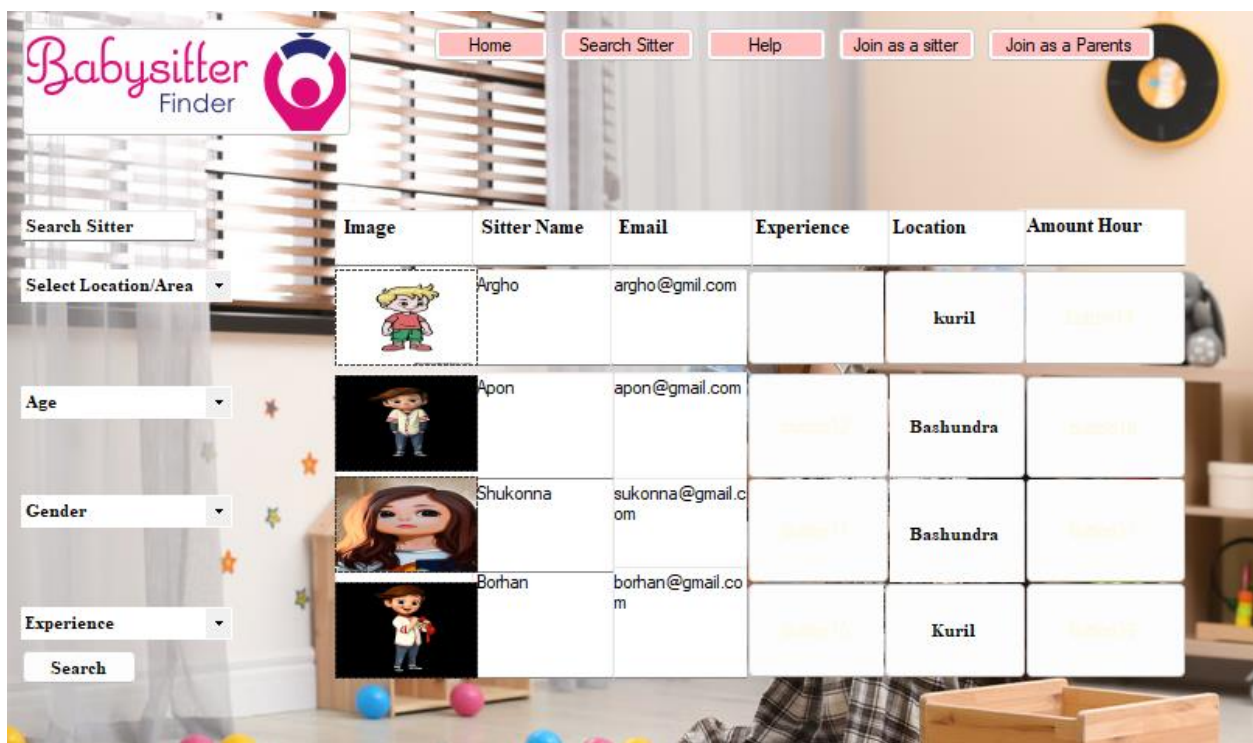 Additionally, parents furnish details such as the father's name, mother's name, baby's name, and the total number of babies in the family. This information helps create personalized profiles within the system. Parents have the ability to browse through a list of available babysitters within the system and book them based on their preferences and requirements. Each babysitter has a unique profile associated with them. .The babysitter has their own profile with unique id ,name , email, phone no , age , gender , experience etc. The babysitter takecare of the baby according to their appointment from the parents. The babysitters works under the name of the Organization .The organization is identified with organization id . The system also stores organizations' name ,address contact and email .

ER Diagram :
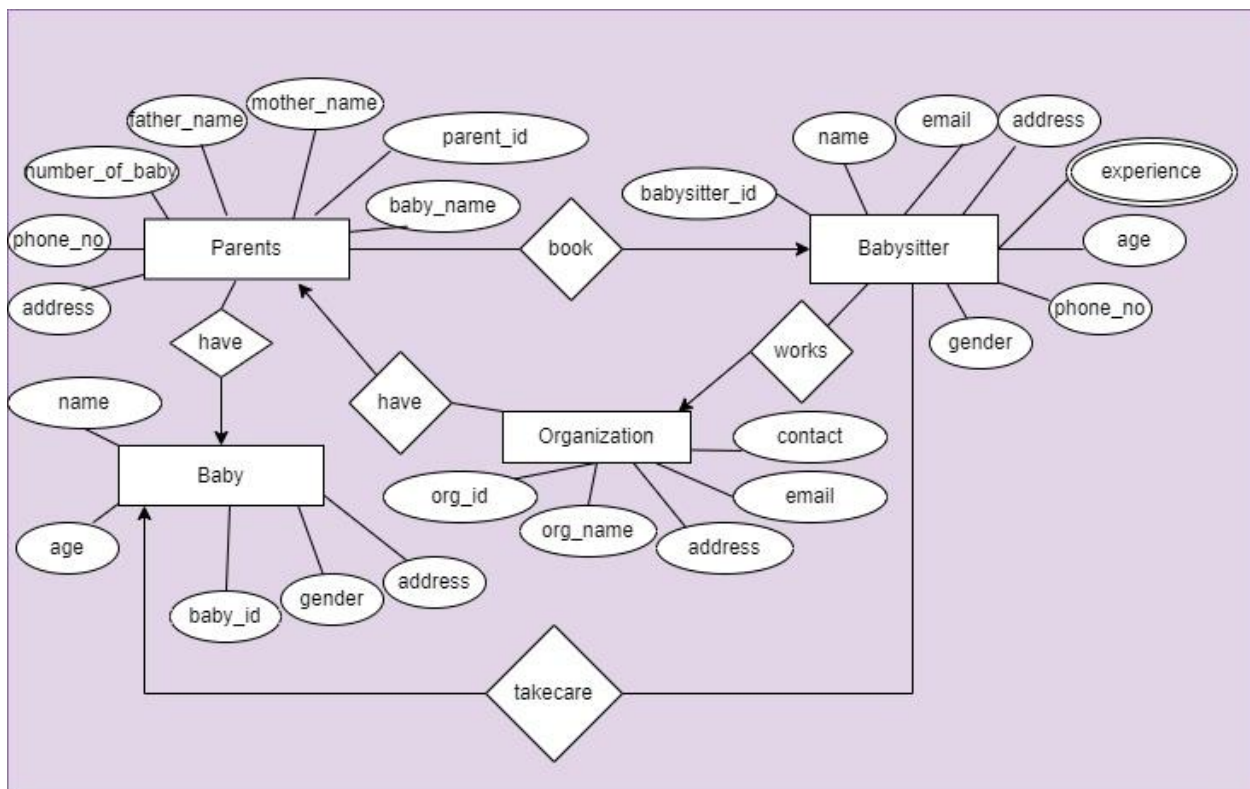
Normalization :

Normalization:
Have
UNF
have(P_i'd, baby-name, mother_name, father_name, phone_no, address, B_i'd, name, email, address, phone_no, age, experience, gender).
1NF
There is no multivalued attribute.
1. Parents (P_i'd**,** baby-name, mother_name, father_name, phone_no, B_i'd, name, email, address, phone_no, age, experience, gender).

2NF
1. P_i'd, baby-name, mother_name, father_name, phone_no, address.
2. B_i'd, name, email, phone_no, age, experience, gender.

3NF
1. P_i'd, baby-name, mother_name, father_name.
2. phone_no, address.
3. B_i'd, name.
4. email, phone_no, age, experience, gender

Table creation
1. P_i'd**,** baby-name, mother_name, father_name.
2. phone_no, addressa, a_i'd.
3. B_i'd, name, **a_id**.
4. email, phone_no, age, experience, gender, **P_i'd.**
5. B_i'd, P_i'd.

BOOK
UNF
Book(P_i'd, baby-name, mother_name, father_name, phone_no, address, babysitter_id, name, email, address, phone_no, age, experience, gender)
1NF
Experience is a multi valued attribute.
1. P_i'd, baby-name, mother_name, father_name, phone_no, address, babysitter_id, name, email, address, phone_no, age, experience, gender.

2NF
1. P_i'd, baby-name, mother_name, father_name, phone_no, address.
2. babysitter_id, name, email, address, phone_no, age, experience, gender.

3NF
1. P_i'd, baby-name, mother_name, father_name.
2. phone_no, address.
3. babysitter_id, name.

4. name, email, address, phone_no, age, experience, gender.

Table Creation

1. P_i'd, baby-name, mother_name, father_name.

2. phone_no, address, a_i'd.

3. babysitter_id, name, **P_i'd.**

4. name, email, address, phone_no, age, experience, gender, **a_i'd, babysitter_id.**

Works
UNF
Works (babysitter_id, name, email, address, phone_no, age, experience, gender, org_id, org_name, contact, email, address)
1NF
Experience is a multi valued attribute.
1. babysitter_id, name, email, address, phone_no, age, experience, gender, org_id, org_name, contact, email, address.

2NF
1. babysitter_id, name, email, address, phone_no, age, experience, gender.
2. org_i'd, org_name, contact, email, address.

3NF
1. babysitter_id, name.
2. email, address, phone_no, age, experience, gender.
3. org_i'd, org_name,
4. contact, email, address.

Table creation
1. babysitter_id, name.
2. email, address, phone_no, age, experience, gender, a_i'd.
3. org_i'd, org_name,
4. contact, email, address, **babysitter_id**.
5. a_i'd, org_id.

Have(organization)
UNF
Have(org_i'd, org_neme ,contact, email, address, P_i'd, baby_name, mother_name, father_name, phone_no, address).
1NF
There is no multi valued attribute.
1. org_i'd, org_name, contact, email, address, P_i'd, baby_name, mother_name, father_name, phone_no.

2NF
1. org_i'd, org_name, contact, email, address.

2. P_i'd, baby_name, mother_name, father_name, number_of_baby, phone_no.

3NF
1. org_i'd, org_name.
2. contact, email, address.
3. P_i'd, baby_name, mother_name, father_name.

Table creation
1. org_i'd, org_name.
2. contact, email, address, a_i'd.
3. P_i'd, baby_name, mother_name, father_name, **org_id**.
4. **a_i'd**, **org_i'd**.

Take Care
UNF
Take care (babysitter_id, name, email, address, phone_no, age, experience, gender, B_i'd, name, age, gender, address)
1NF
Experience is a multi valued attribute.
1. babysitter_id, name, email, address, phone_no, age, experience, gender, B_i'd, name, age, gender, address.

2NF
1. babysitter_id, name, email, address, phone_no, age, experience, gender.
2. B_i'd, name, age, gender, address.

3NF
1. babysitter_id, name.
2. email, address, phone_no, age, experience, gender.
3. B_i'd, name.
4. age, gender, address.

Table Creation
1. babysitter_id, name.
2. email, address, phone_no, age, experience, gender, a_i'd.
3. B_i'd, name.
4. age, gender, address, **a_i'd**.
5. **babysitter_id, B_i'd.**

Temporary Table
1. ~~P_i'd, baby-name, mother_name, father_name.~~
2. ~~phone_no, addressa, a_i'd.~~
3. B_i'd, name, **a_id**.
4. ~~email, phone_no, age, experience, gender, **P_i'd.**~~
5. **B_i'd, P_i'd.**
6. P_i'd, baby-name, mother_name, father_name.
7. phone_no, address, a_i'd.
8. babysitter_id, name, **P_i'd.**
9. name, email, address, phone_no, age, experience, gender, **a_i'd, babysitter_id.**
10. ~~babysitter_id, name.~~
11. ~~email, address, phone_no, age, experience, gender, a_i'd.~~
12. ~~org_i'd, org_name,~~
13. contact, email, address, **babysitter_id**.
14. **a_i'd, org_id.**
15. org_i'd, org_name.
16. ~~contact, email, address, a_i'd.~~
17. ~~P_i'd, baby_name, mother_name, father_name, **org_id**.~~
18. **a_i'd**, **org_i'd**.
19. babysitter_id, name.
20. email, address, phone_no, age, experience, gender, a_i'd.
21. ~~B_i'd, name.~~
22. age, gender, address, **a_i'd**.
23. ~~babysitter_id, B_i'd.~~

Final Table
1. B_i'd, name, **a_id**.
2. **B_i'd, P_i'd.**
3. P_i'd, baby-name, mother_name, father_name.
4. phone_no, address, a_i'd.
5. babysitter_id, name, **P_i'd.**
6. name, email, address, phone_no, age, experience, gender, **a_i'd, babysitter_id.**
7. contact, email, address, **babysitter_id**.
8. **a_i'd, org_id.**
9. org_i'd, org_name.
10. babysitter_id, name.
11. email, address, phone_no, age, experience, gender, a_i'd.
12. age, gender, address, **a_i'd**.

Schema Diagram :

Table Creation :

##############################
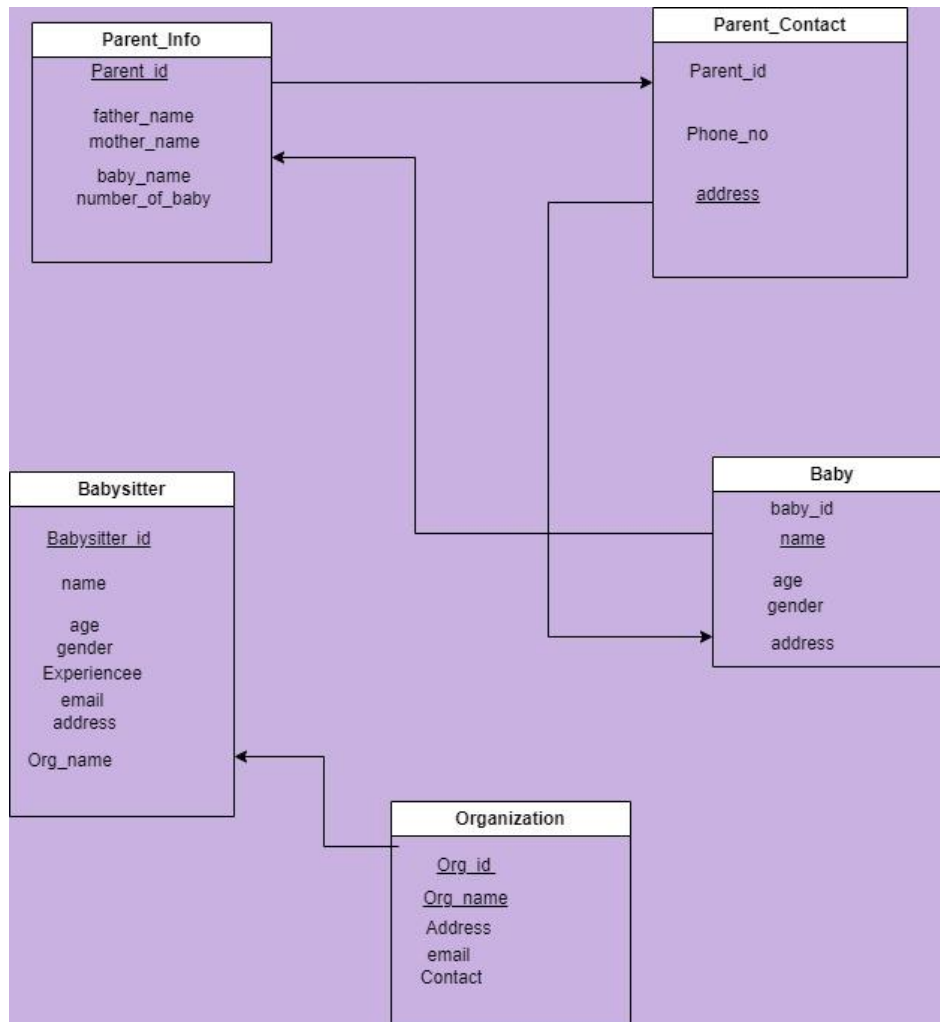
```sql
CREATE TABLE Baby (
    Name VARCHAR(50) Primary key,

    Baby_id  VARCHAR(10) Primary Key,
    Age INT,
    Gender VARCHAR(10),
    Address VARCHAR(100)
);
```
###########################

```sql
CREATE TABLE Parent_Info (

Parent_id VARCHAR(50) Primary key,

 Father_name VARCHAR(50),

 Mother_name VARCHAR(50),

 Number_of_baby INT,

 Baby_name VARCHAR(50),

 FOREIGN KEY (Baby_name) REFERENCES Baby(Name)

);
```

####################################

```sql
CREATE TABLE Parent_Contact(

Parent_id VARCHAR (50),

 Phone_no VARCHAR(15),

 Address VARCHAR(100),

FOREIGN KEY (Parent_id) REFERENCES Parent_info(Parent_id)

);
```

#############################

```sql
CREATE TABLE Organization (

    Org_name VARCHAR(20) PRIMARY KEY,
```

```sql
    Org_id INT PRIMARY KEY,

    Email VARCHAR(255),

    Address VARCHAR(255),

    Contact VARCHAR(20)

);
```

##################################

```sql
CREATE TABLE babysitter (

Babysitter_id VARCHAR(20) PRIMARY KEY,
  Name VARCHAR(50),
  Age INT,
  Gender VARCHAR(10),
  Email VARCHAR(100),
  Address VARCHAR(200),
  PhoneNo VARCHAR(20),
  Experience INT

FOREIGN KEY (Email) REFERENCES Organization(Email)


);
```

Data Insertion :

```sql
INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1000, 'Partha', 7, 'Male', 'Kuril');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1001, 'Argho', 6, 'Male', 'Kuril');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1002, 'Sukonna', 5, 'Female', 'Bashundhora');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1003, 'Apon', 4, 'Male', 'Bisshoroad');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1004, 'Anita', 7, 'Female', 'Dhanmondi');
```

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1005, 'Madhobi', 4, 'Female', 'Kuril');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1006, 'Ahana', 4, 'Female', 'Sadarghat');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1007, 'Rafi', 3, 'Male', 'Cantonment');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1008, 'Tamim', 5, 'Male', 'Dhanmondi');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1009, 'Akil', 5, 'Male', 'Gulsan RA/A');

INSERT INTO Baby (Baby_id, Name, Age, Gender, Address)

VALUES (1010, 'Aklima', 2, 'Female', 'Nikunjo -2');

SELECT * FROM Baby;

DESCRIBE Baby;

NAME	BABY_ID	AGE	GENDER	ADDRESS
Partha	1000	7	Male	Kuril
Argho	1001	6	Male	Kuril
Sukonna	1002	5	Female	Bashundhora
Apon	1003	4	Male	Bisshoroad
Anita	1004	7	Female	Dhanmondi
Madhobi	1005	4	Female	Kuril
Ahana	1006	4	Female	Sadarghat
Rafi	1007	3	Male	Cantonment
Tamim	1008	5	Male	Dhanmondi
Akil	1009	5	Male	Gulsan RA/A
Aklima	1010	2	Female	Nikunjo -2

11 rows returned in 0.00 seconds   CSV Export

Object Type **TABLE** Object **BABY**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BABY	NAME	Varchar2	50	-	-	1	-	-	-
	BABY_ID	Varchar2	10	-	-	-	✓	-	-
	AGE	Number	-	-	0	-	✓	-	-
	GENDER	Varchar2	10	-	-	-	✓	-	-
	ADDRESS	Varchar2	100	-	-	-	✓	-	-
									1 - 5

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2010,'Romesh', 'Ahana', 2, 'Partha');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2011,'Tomesh', 'Himi Roy', 2,'Argho');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2012,'Shohorab', 'Rehena', 2,'Sukonna');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2013,'Abdul Malek', 'Rehena', 2, 'Apon');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2014,'Ali', 'Shokhina', 2, 'Anita');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2015,'Siam hauq', 'Khusi begum', 3, 'Madhobi');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2016,'Pomesh', 'Gita Rani', 2, 'Ahana');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2017,'Karim', 'Purnima', 2, 'Rafi');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2018,'Rahim', 'Moushomi', 3, 'Tamim');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2019,'Lokman', 'Popi', 2, 'Akil');

INSERT INTO Parent_info (Parent_id , Father_name, Mother_name, Number_of_baby, Baby_name)

VALUES (2020,'Rohit', 'Mousi', 1, 'Aklima');

SELECT * FROM Parent_Info
DESCRIBE Parent_Info;

PARENT_ID	FATHER_NAME	MOTHER_NAME	NUMBER_OF_BABY	BABY_NAME
2010	Romesh	Ahana	2	Partha
2011	Tomesh	Himi Roy	2	Argho
2012	Shohorab	Rehena	2	Sukonna
2013	Abdul Malek	Rehena	2	Apon
2014	Ali	Shokhina	2	Anita
2015	Siam hauq	Khusi begum	3	Madhobi
2016	Pomesh	Gita Rani	2	Ahana
2017	Karim	Purnima	2	Rafi
2018	Rahim	Moushomi	3	Tamim
2019	Lokman	Popi	2	Akil
2020	Rohit	Mousi	1	Aklima

11 rows returned in 0.07 seconds   CSV Export

Object Type **TABLE** Object **PARENT_INFO**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PARENT_INFO	PARENT_ID	Varchar2	50	-	-	1	-	-	-
	FATHER_NAME	Varchar2	50	-	-	-	✓	-	-
	MOTHER_NAME	Varchar2	50	-	-	-	✓	-	-
	NUMBER_OF_BABY	Number	-	-	0	-	✓	-	-
	BABY_NAME	Varchar2	50	-	-	-	✓	-	-
									1 - 5

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2010,01717486552,'Kuril');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2011,01717486352,'Kuril');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2012,01717486550,'Bashundhara');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2013,01717486532,'Bissoroad');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2014,01717496552,'Dhanmondi');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2015,01717455552,'Kuril');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2016,01717486588,'Sadarghat');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2017,01717486662,'Cantonment');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2018,01717422252,'Dhanmondi');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2019,01727486552,'Gulshan RA/A');

INSERT INTO Parent_contact(Parent_id, Phone_no, Address)

VALUES (2020,01774865520,'Nikunja-2');

SELECT * FROM Parent_contact

DESCRIBE Parent_contact;

PARENT_ID	PHONE_NO	ADDRESS
2010	1717486552	Kuril
2011	1717486352	Kuril
2012	1717486550	Bashundhara
2013	1717486532	Bissoroad
2014	1717496552	Dhanmondi
2014	1717496552	Dhanmondi
2015	1717455552	Kuril
2016	1717486588	Sadarghat
2017	1717486662	Cantonment
2018	1717422252	Dhanmondi
2019	1727486552	Gulshan RA/A
2020	1774865520	Nikunja-2

12 rows returned in 0.02 seconds   CSV Export

Object Type TABLE Object PARENT_CONTACT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PARENT_CONTACT	PARENT_ID	Varchar2	50	-	-	-	✓	-	-
	PHONE_NO	Varchar2	15	-	-	-	✓	-	-
	ADDRESS	Varchar2	100	-	-	-	✓	-	-
									1 - 3

INSERT INTO babysitter (Babysitter_id,Name, Age, Gender, Email, Address, Phone_no, Experience)

VALUES (3001,'Alifa', 25, 'Female', 'BabySitfinder@gmail.com', 'Rampura', 0174567890, 3);

INSERT INTO babysitter (Babysitter_id,Name, Age, Gender, Email, Address, Phone_no, Experience)

VALUES (3002,'Mimi', 22, 'Female', 'BabySitfinder@gmail.com', 'Mirpur', 0174567879, 3);

INSERT INTO babysitter (Babysitter_id,Name, Age, Gender, Email, Address, Phone_no, Experience)

VALUES (3003,'Khalid', 24, 'Male', 'BabySitfinder@gmail.com', 'Gulshan', '01945678790', 3);

INSERT INTO babysitter (Babysitter_id,Name, Age, Gender, Email, Address, Phone_no, Experience)

VALUES (3004,'Rubi', 26, 'Female', 'BabySitfinder@gmail.com', 'Bashundhora', 0174567860, 2);

INSERT INTO babysitter (Babysitter_id,Name, Age, Gender, Email, Address, Phone_no, Experience)

VALUES (3005,'Momo', 21, 'Female', 'BabySitfinder@gmail.com', 'Mirpur', 0174567279, 3);

SELECT * FROM babysitter;

DESCRIBE babysitter;

BABYSITTER_ID	NAME	AGE	GENDER	EMAIL	ADDRESS	PHONE_NO	EXPERIENCE
3001	Alifa	25	Female	BabySitfinder@gmail.com	Rampura	174567890	3
3002	Mimi	22	Female	BabySitfinder@gmail.com	Mirpur	174567879	3
3003	Khalid	24	Male	BabySitfinder@gmail.com	Gulshan	01945678790	3
3004	Rubi	26	Female	BabySitfinder@gmail.com	Bashundhora	174567860	2
3005	Momo	21	Female	BabySitfinder@gmail.com	Mirpur	174567279	3

5 rows returned in 0.04 seconds   CSV Export

Object Type **TABLE** Object **BABYSITTER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BABYSITTER	BABYSITTER_ID	Varchar2	20	-	-	1	-	-	-
	NAME	Varchar2	50	-	-	-	✓	-	-
	AGE	Number	-	-	0	-	✓	-	-
	GENDER	Varchar2	10	-	-	-	✓	-	-
	EMAIL	Varchar2	100	-	-	-	✓	-	-
	ADDRESS	Varchar2	200	-	-	-	✓	-	-
	PHONE_NO	Varchar2	20	-	-	-	✓	-	-
	EXPERIENCE	Number	-	-	0	-	✓	-	-
									1 - 8

INSERT INTO Organization (Org_name, Org_id, Email, Address, Contact)

VALUES ('BabySitFinder.Com', 255876, 'BabySitfinder@gmail.com', 'Dhanmondi', '09567896666');

SELECT * FROM Organization

DESCRIBE Organization;

ORG_NAME	ORG_ID	EMAIL	ADDRESS	CONTACT
BabySitFinder.Com	255876	BabySitfinder@gmail.com	Dhanmondi	09567896666

1 rows returned in 0.03 seconds   CSV Export

Object Type **TABLE** Object **ORGANIZATION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORGANIZATION	ORG_NAME	Varchar2	20	-	-	-	✓	-	-
	ORG_ID	Number	-	-	0	-	✓	-	-
	EMAIL	Varchar2	255	-	-	1	-	-	-
	ADDRESS	Varchar2	255	-	-	-	✓	-	-
	CONTACT	Varchar2	20	-	-	-	✓	-	-
									1 - 5

Query Writing :

Single-Row Functions

Question1: Retrieve the length of each father's name in the "parent_info" table.

SELECT Father_name, LENGTH(Father_name) AS Namelength FROM Parent_info;

FATHER_NAME	NAMELENGTH
Romesh	6
Tomesh	6
Shohorab	8
Abdul Malek	11
Ali	3
Siam hauq	9
Pomesh	6
Karim	5
Rahim	5
Lokman	6
Rohit	5

11 rows returned in 0.04 seconds   CSV Export

Question2: Convert the ages of babies in the "Baby" table from years to months.

SELECT Name,Baby_id Age, Age * 12 AS AgeInMonths FROM Baby;

NAME	AGE	AGEINMONTHS
Partha	1000	84
Argho	1001	72
Sukonna	1002	60
Apon	1003	48
Anita	1004	84
Madhobi	1005	48
Ahana	1006	48
Rafi	1007	36
Tamim	1008	60
Akil	1009	60
Aklima	1010	24

11 rows returned in 0.04 seconds   CSV Export

Question3: Retrieve the uppercase version of the Father_name in the "Parent_info " table.

SELECT Mother_name, UPPER(Father_name) AS Uppercase_Father_name
FROM Parent_info;

MOTHER_NAME	UPPERCASE_FATHER_NAME
Ahana	ROMESH
Himi Roy	TOMESH
Rehena	SHOHORAB
Rehena	ABDUL MALEK
Shokhina	ALI
Khusi begum	SIAM HAUQ
Gita Rani	POMESH
Purnima	KARIM
Moushomi	RAHIM
Popi	LOKMAN
Mousi	ROHIT

11 rows returned in 0.02 seconds   CSV Export

Group Functions:

Question1: Calculate the average age of babies in the "Baby" table.

SELECT AVG(Age) AS Average_Age FROM Baby;

AVERAGE_AGE
4.727272727272727272727272727272727272727273

1 rows returned in 0.07 seconds   CSV Export

Question2: Find the minimum and maximum ages among the Babies in the "Baby" table.

SELECT MIN(Age) AS Minimum_Age, MAX(Age) AS Maximum_Age FROM Baby;

MINIMUM_AGE	MAXIMUM_AGE
2	7

1 rows returned in 0.05 seconds   CSV Export

Question3: Count the number of parents in the "Parent_Info" table.

SELECT COUNT(*) AS Total_Parents FROM Parent_info;

TOTAL_PARENTS
11

1 rows returned in 0.03 seconds   CSV Export

Subqueries

Question1: Retrieve the details of parents whose babies are younger than 2 years.

SELECT Father_name, Mother_name, Number_Of_Baby
FROM Parent_info
WHERE Baby_name IN (SELECT Name FROM Baby WHERE Age < 5);

FATHER_NAME	MOTHER_NAME	NUMBER_OF_BABY
Abdul Malek	Rehena	2
Siam hauq	Khusi begum	3
Pomesh	Gita Rani	2
Karim	Purnima	2
Rohit	Mousi	1

5 rows returned in 0.04 seconds   CSV Export

Question2: Find the fathers who have more babies than the average number of babies.

SELECT Father_name, Number_Of_Baby
FROM Parent_Info
WHERE Number_Of_Baby > (SELECT AVG(Number_Of_Baby) FROM Parent_info);

FATHER_NAME	NUMBER_OF_BABY
Siam hauq	3
Rahim	3

2 rows returned in 0.06 seconds   CSV Export

Question 03:Retrieve the babysitters' information whose address matches any of the addresses in the "Organization" table.

SELECT Name, Age, Address, Phone_no
FROM babysitter
WHERE Address IN (SELECT Address FROM Organization);

no data found

Joining

Question1: Retrieve the details of parents along with their corresponding baby details.

SELECT p.Father_name, p.Mother_name, p.Number_Of_Baby, b.Age, b.Gender
FROM Parent_info p
INNER JOIN Baby b ON p.Baby_name= b.Name;

FATHER_NAME	MOTHER_NAME	NUMBER_OF_BABY	AGE	GENDER
Romesh	Ahana	2	7	Male
Tomesh	Himi Roy	2	6	Male
Shohorab	Rehena	2	5	Female
Abdul Malek	Rehena	2	4	Male
Ali	Shokhina	2	7	Female
Siam hauq	Khusi begum	3	4	Female
Pomesh	Gita Rani	2	4	Female
Karim	Purnima	2	3	Male
Rahim	Moushomi	3	5	Male
Lokman	Popi	2	5	Male
Rohit	Mousi	1	2	Female

11 rows returned in 0.03 seconds   CSV Export

Question 02: Find the babies who have the same address as the parents.
SELECT b.Baby_id, b.Name, b.Age, b.Gender, b.Address
FROM Baby b
INNER JOIN Parent_contact p ON b.Address = p.Address;

BABY_ID	NAME	AGE	GENDER	ADDRESS
1005	Madhobi	4	Female	Kuril
1001	Argho	6	Male	Kuril
1000	Partha	7	Male	Kuril
1005	Madhobi	4	Female	Kuril
1001	Argho	6	Male	Kuril
1000	Partha	7	Male	Kuril
1008	Tamim	5	Male	Dhanmondi
1004	Anita	7	Female	Dhanmondi
1008	Tamim	5	Male	Dhanmondi
1004	Anita	7	Female	Dhanmondi
1005	Madhobi	4	Female	Kuril
1001	Argho	6	Male	Kuril
1000	Partha	7	Male	Kuril
1006	Ahana	4	Female	Sadarghat
1007	Rafi	3	Male	Cantonment

More than 15 rows available. Increase rows selector to view more rows.

15 rows returned in 0.04 seconds   CSV Export

Question3: Retrieve the details along with the names of the organizations where they work.
SELECT p.Address, o.Org_name, o.Email, o.Contact
FROM Parent_contact p
INNER JOIN Organization o ON p.Address = o.Address;

ADDRESS	ORG_NAME	EMAIL	CONTACT
Dhanmondi	BabySitFinder.Com	BabySitfinder@gmail.com	09567896666
Dhanmondi	BabySitFinder.Com	BabySitfinder@gmail.com	09567896666
Dhanmondi	BabySitFinder.Com	BabySitfinder@gmail.com	09567896666

3 rows returned in 0.08 seconds   CSV Export

Views

 Question1: Create a view named "ParentsView" that displays the details of parents.

CREATE VIEW ParentsView AS

SELECT Parent_id ,Father_name, Mother_name, Number_Of_Baby, baby_name

FROM Parent_info;

```
View created.
```

```
0.00 seconds
```

Question2: Retrieve the details of parents using the "ParentsView" view.

SELECT * FROM ParentsView;

PARENT_ID	FATHER_NAME	MOTHER_NAME	NUMBER_OF_BABY	BABY_NAME
2010	Romesh	Ahana	2	Partha
2011	Tomesh	Himi Roy	2	Argho
2012	Shohorab	Rehena	2	Sukonna
2013	Abdul Malek	Rehena	2	Apon
2014	Ali	Shokhina	2	Anita
2015	Siam hauq	Khusi begum	3	Madhobi
2016	Pomesh	Gita Rani	2	Ahana
2017	Karim	Purnima	2	Rafi
2018	Rahim	Moushomi	3	Tamim
2019	Lokman	Popi	2	Akil
2020	Rohit	Mousi	1	Aklima

11 rows returned in 0.03 seconds   CSV Export

Question3: Modify the "ParentsView" to include only parents with more than one baby.

CREATE OR REPLACE VIEW ParentsView AS
SELECT Parent_id,Father_name, Mother_name, Number_Of_Baby, baby_name
FROM Parent_info
WHERE Number_Of_Baby > 1;

```
View created.
```

```
0.03 seconds
```

Synonyms

Question1: Create a synonym named "Parent_info_Synonym" for the "parents" table.

CREATE SYNONYM Parent_info_Synonym FOR Parent_info;

```
Synonym created.
```

0.00 seconds

Question2: Retrieve the details of parents using the "Parent_info_Synonym" synonym.

SELECT * FROM Parent_info_Synonym;

PARENT_ID	FATHER_NAME	MOTHER_NAME	NUMBER_OF_BABY	BABY_NAME
2010	Romesh	Ahana	2	Partha
2011	Tomesh	Himi Roy	2	Argho
2012	Shohorab	Rehena	2	Sukonna
2013	Abdul Malek	Rehena	2	Apon
2014	Ali	Shokhina	2	Anita
2015	Siam hauq	Khusi begum	3	Madhobi
2016	Pomesh	Gita Rani	2	Ahana
2017	Karim	Purnima	2	Rafi
2018	Rahim	Moushomi	3	Tamim
2019	Lokman	Popi	2	Akil
2020	Rohit	Mousi	1	Aklima

|1 rows returned in 0.09 seconds   CSV Export

Question3: Drop the synonym "Parent_info_Synonym".

DROP SYNONYM Parent_info_Synonym;


```
Synonym dropped.
```

0.00 seconds

PL/SQL

Procedure:

Question1:

Create a PL/SQL procedure named IncreaseBabyAge that takes in a Baby_id and an AgeIncrease as input parameters and updates the Baby table by increasing the age of the specified baby by the given number of years.

```
CREATE OR REPLACE PROCEDURE IncreaseBabyAge(
    p_Baby_id IN Baby.Baby_id%TYPE,
    p_AgeIncrease IN Baby.Age%TYPE
) AS
BEGIN
    UPDATE Baby
    SET Age = Age + p_AgeIncrease
    WHERE Baby_id = p_Baby_id;

    COMMIT;
END;
/
```

```
Procedure created.
```

Question02: Create a procedure to increase the age of babies by 2 years .

```
DECLARE
    v_Result VARCHAR2(100);

 -- Variable to capture the result
BEGIN
    IncreaseBabyAge(p_Baby_id => 123, p_AgeIncrease => 2);
    v_Result := 'Age increased successfully';
    DBMS_OUTPUT.PUT_LINE(v_Result);
EXCEPTION
    WHEN OTHERS THEN
        v_Result := 'Error: ' || SQLERRM;
        DBMS_OUTPUT.PUT_LINE(v_Result);
END;
/
```

```
Age increased successfully

Statement processed.


0.11 seconds
```

Question03:Create a procedure to update the address of first 2 babies .

```
CREATE OR REPLACE PROCEDURE UpdateBabyAddresses AS
BEGIN
    -- Update the address for Baby 1
    UPDATE Baby
    SET Address = 'Badda'
    WHERE Baby_id = 1;

    DBMS_OUTPUT.PUT_LINE('Address for Baby 1 has been updated to Badda.');

    -- Update the address for Baby 2
    UPDATE Baby
    SET Address = 'Badda'
    WHERE Baby_id = 2;

    DBMS_OUTPUT.PUT_LINE('Address for Baby 2 has been updated to Badda.');

    COMMIT;
END;
/

BEGIN
    UpdateBabyAddresses;
END;
/
```

```
Address for Baby 1 has been updated to Badda.
Address for Baby 2 has been updated to Badda.

Statement processed.
```

Function:

Question 1: FunctionHow can you create a PL/SQL function to retrieve the Email of a babysitter based on their Babysitter_id?

```
CREATE OR REPLACE FUNCTION get_babysitter_email(p_babysitter_id IN NUMBER)
RETURN VARCHAR2 IS
   v_email VARCHAR2(100);
BEGIN
   SELECT email INTO v_email
   FROM babysitter
   WHERE babysitter_id = p_babysitter_id;

   RETURN v_email;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL; -- Return NULL if babysitter not found
   WHEN OTHERS THEN
      -- Handle other exceptions here
      RAISE;
END;
```

Function created.

0.06 seconds

/

Question 2: Write a PL/SQL function to update the Address of a babysitter based on their
Babysitter_id.

```
CREATE OR REPLACE FUNCTION update_babysitter_address(
   p_babysitter_id IN NUMBER,
   p_new_address IN VARCHAR2
) RETURN BOOLEAN IS
BEGIN
   UPDATE babysitter
   SET address = p_new_address
   WHERE babysitter_id = p_babysitter_id;



   IF SQL%ROWCOUNT > 0 THEN
      dbms_output.put_line('Address updated successfully.');
      RETURN TRUE; -- Update successful
   ELSE
      dbms_output.put_line('No rows updated. Babysitter not found.');
      RETURN FALSE; -- No rows updated, babysitter_id not found
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      dbms_output.put_line('An error occurred: ' || SQLERRM);
```

```
      RAISE;
END;
/

DECLARE
   v_result BOOLEAN;
BEGIN
   v_result := update_babysitter_address(p_babysitter_id => 123, p_new_address => '123 Main
St');
   IF v_result THEN
      dbms_output.put_line('Update successful.');
   ELSE
      dbms_output.put_line('Update failed.');
   END IF;
END;
/
```

```
No rows updated. Babysitter not found.
Update failed.

Statement processed.


0.10 seconds
```

Question 3: How can you create a PL/SQL function to calculate the average Age of all
babysitters of a specific Gender?

```
CREATE OR REPLACE FUNCTION calculate_avg_age_by_gender(p_gender IN VARCHAR2) RETURN
NUMBER IS
   v_avg_age NUMBER;
BEGIN
   SELECT AVG(age) INTO v_avg_age
   FROM babysitter
   WHERE gender = p_gender;

   IF v_avg_age IS NOT NULL THEN
      dbms_output.put_line('Average age for gender ' || p_gender || ': ' || v_avg_age);
   ELSE
      dbms_output.put_line('No babysitters found with gender ' || p_gender);
   END IF;

   RETURN v_avg_age;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      dbms_output.put_line('No babysitters found with gender ' || p_gender);
```

```
      RETURN NULL; -- Return NULL if no babysitters found with the given gender
   WHEN OTHERS THEN
      dbms_output.put_line('An error occurred: ' || SQLERRM);
      RAISE;
END;
/

DECLARE
   v_average_age NUMBER;
BEGIN
   v_average_age := calculate_avg_age_by_gender('Female');
END;
/
```

```
Average age for gender Female: 23.5

Statement processed.


0.05 seconds
```

Cursor:

Question 1: How can you use a PL/SQL cursor to retrieve the names of all parents along with the number of babies they have?

```
DECLARE
   CURSOR parent_info_cursor IS
      SELECT Father_name, Mother_Name, Number_of_baby
      FROM Parent_info;
BEGIN
   FOR parent_rec IN parent_info_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('Parent: ' || parent_rec.Father_name || ' and ' ||
parent_rec.Mother_Name || ', Babies: ' || parent_rec.Number_of_baby);
   END LOOP;
END;
/
```

```
Parent: Romesh and Ahana, Babies: 2
Parent: Tomesh and Himi Roy, Babies: 2
Parent: Shohorab and Rehena, Babies: 2
Parent: Abdul Malek and Rehena, Babies: 2
Parent: Ali and Shokhina, Babies: 2
Parent: Siam hauq and Khusi begum, Babies: 3
Parent: Pomesh and Gita Rani, Babies: 2
Parent: Karim and Purnima, Babies: 2
Parent: Rahim and Moushomi, Babies: 3
Parent: Lokman and Popi, Babies: 2
Parent: Rohit and Mousi, Babies: 1

Statement processed.


0.11 seconds
```

Question 2: Write a PL/SQL cursor that updates the Father_name of a parent based on their Parent_id.

```
DECLARE
    v_parent_id_to_update NUMBER := 123; -- Replace with the desired Parent_id
    v_new_father_name VARCHAR2(100) := 'New Father Name';
BEGIN
    UPDATE Parent_info
    SET Father_name = v_new_father_name
    WHERE Parent_id = v_parent_id_to_update;

    IF SQL%ROWCOUNT > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Father name updated successfully.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Parent not found.');
    END IF;
END;
/
```

```
Parent not found.

1 row(s) updated.


0.10 seconds
```

Question 3: Create a PL/SQL cursor to calculate and display the average number of babies across all parents.

```
DECLARE
    v_total_babies NUMBER := 0;
    v_total_parents NUMBER := 0;
    v_avg_babies NUMBER;
```

```
BEGIN
    FOR parent_rec IN (SELECT Number_of_baby FROM Parent_info) LOOP
        v_total_babies := v_total_babies + parent_rec.Number_of_baby;
        v_total_parents := v_total_parents + 1;
    END LOOP;

    IF v_total_parents > 0 THEN
        v_avg_babies := v_total_babies / v_total_parents;
        DBMS_OUTPUT.PUT_LINE('Average number of babies per parent: ' || v_avg_babies);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No parents found.');
    END IF;
END;
/
```

```
Average number of babies per parent: 2.0909090909090909090909090909090909090909

Statement processed.

0.06 seconds
```

.

Record :

Question 1: How can you use a PL/SQL record to retrieve and display the details of a baby based on their Baby_id?

```
DECLARE
    v_baby_details Baby%ROWTYPE;
BEGIN
    SELECT * INTO v_baby_details
    FROM Baby
    WHERE Baby_id = 123; -- Replace with the desired Baby_id

    DBMS_OUTPUT.PUT_LINE('Baby ID: ' || v_baby_details.Baby_id);
    DBMS_OUTPUT.PUT_LINE('Age: ' || v_baby_details.Age);
    DBMS_OUTPUT.PUT_LINE('Gender: ' || v_baby_details.Gender);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found for the given Baby_id.');
END;
/
```

Question 2: Write a PL/SQL block that updates the Age of a baby based on their Baby_id.

```
DECLARE
    v_baby_id_to_update NUMBER := 456; -- Replace with the desired Baby_id
    v_new_age NUMBER := 12; -- Replace with the desired new Age
BEGIN
    UPDATE Baby
    SET Age = v_new_age
    WHERE Baby_id = v_baby_id_to_update;

    IF SQL%ROWCOUNT > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Age updated successfully.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Baby not found.');
    END IF;
END;
/
```

Question 3:Create a PL/SQL block that calculates and displays the count of babies for each gender.

```
DECLARE
    TYPE gender_count_rec IS RECORD (
        gender VARCHAR2(10),
        baby_count NUMBER
    );

    v_gender_count gender_count_rec;
BEGIN
    FOR gender_rec IN (SELECT DISTINCT Gender FROM Baby) LOOP
        SELECT gender_rec.Gender, COUNT(*) INTO v_gender_count
        FROM Baby
        WHERE Gender = gender_rec.Gender;
```

```
      DBMS_OUTPUT.PUT_LINE('Gender: ' || v_gender_count.gender || ', Baby Count: ' ||
v_gender_count.baby_count);
   END LOOP;
END;
/
```

```
Gender: Male, Baby Count: 6
Gender: Female, Baby Count: 5

Statement processed.


0.13 seconds
```

Trigger:

Question 1: How can you create a PL/SQL trigger that automatically updates the Number_of_baby column when a new baby is added for a parent?

```
CREATE OR REPLACE TRIGGER update_baby_count
AFTER INSERT ON Baby
FOR EACH ROW
BEGIN
   UPDATE Parent_info
   SET Number_of_baby = Number_of_baby + 1
   WHERE Parent_id = Parent_id;
END;
/
```

```
Trigger created.


0.06 seconds
```

Package :

Question01:Create a PL/SQL package named "BabyManagement" that includes a procedure to insert a new baby's information into the "Baby" table. The "Baby" table has the following columns: Babysitter_id, Name, Age, Gender, Email, Address, Phone_no, and Experience. Write the code to implement this procedure.

```
CREATE OR REPLACE PACKAGE BabyManagement AS
   PROCEDURE Insert Baby(
      p_Babysitter_id IN NUMBER,
```

```
      p_Name IN VARCHAR2,
      p_Age IN NUMBER,
      p_Gender IN VARCHAR2,
      p_Email IN VARCHAR2,
      p_Address IN VARCHAR2,
      p_Phone_no IN VARCHAR2,
      p_Experience IN VARCHAR2
   );
END BabyManagement;
```

Package created.

0.01 seconds

Question02:Create a PL/SQL package named "ParentManagement" that includes a procedure to insert new parent information into the "Parent_info" table. The "Parent_info" table has the following columns: Parent_id, Father_name, Mother_name, Number_of_baby, and Baby_name. Write the code to implement this procedure.

```
CREATE OR REPLACE PACKAGE BODY ParentManagement AS
   PROCEDURE InsertParent(
      p_Parent_id IN NUMBER,
      p_Father_name IN VARCHAR2,
      p_Mother_name IN VARCHAR2,
      p_Number_of_baby IN NUMBER,
      p_Baby_name IN VARCHAR2
   ) IS
   BEGIN
      INSERT INTO Parent_info (
         Parent_id,
         Father_name,
         Mother_name,
         Number_of_baby,
         Baby_name
      ) VALUES (
         p_Parent_id,
         p_Father_name,
         p_Mother_name,
         p_Number_of_baby,
         p_Baby_name
      );
      COMMIT;
   END InsertParent;
END ParentManagement;
```

```
Package Body created.
```

0.01 seconds

Relational Algebra :

1.Find the baby's name and associated address for babysitters who work for an organization with `org_name` equal to "Babysit finder.com".

Solution:

π(name, address) (**Baby** ⋈ (π(babysitter_id) (π (org_id) (σ (org_name = " Babysit finder.com ", Organization)) ⋈ **Babysitter**)))

2. Find the names of babies who have a father with the name "Romesh" and are taken care of by a babysitter named "Momo".

Solution:

π (name) (**Baby** ⋈ (σ (father_name = "Romesh", Parents) ⋈ (σ (name = "Momo", Babysitter))))

3. Increase the age of all babies by 1 year.

Solution:

Baby ← π(B_i'd, name, age + 1, gender, address) (**Baby**)

4. Retrieve the names and ages of babies along with their babysitter's name, if available.

Solution:

π (Baby.name, Baby.age, Babysitter.name) ⋈ (**Baby** ⋈ (**Parents** ⋈ **Babysitter**))

5. Delete all organizations that have a contact email containing "babysitter.com".

Solution:

Organization ← **Organization** - σ (email LIKE "%babysitter.com%", Organization)

6. Retrieve the names of parents whose babies are taken care of by babysitters younger than 25 years.

Solution:

π (mother_name, father_name) (**Parents** ⋈ (**Baby** ⋈ (σ (age < 25, Babysitter))))

7. Find the names and ages of babies along with their parents' names who live in the city "Kuril".

Solution:

π (Baby.name, Baby.age, Parents.mother_name, Parents.father_name) ⋈ (**Baby** ⋈ (σ(address = "Kuril", Parents)))

Conclusion :

The Babysitter Management System proposal presents an innovative solution to simplify and enhance childcare management for modern families and agencies. By providing a user-friendly platform, comprehensive babysitter pool, and efficient scheduling features, the system aims to revolutionize the way parents and babysitters connect and collaborate. We believe that this project will greatly benefit the childcare community and offer significant convenience to busy families. We look forward to the opportunity to develop and implement the Babysitter Management System. Thank you for considering our proposal, and we are available to discuss any further details or modifications required.