Choose an item.

Choose an item.

AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE C TECHNOLOGY

Advance Database Management System

Final Report

Project Name – Babysitting Management System

Section: B

Supervised By

SIFAT RAHAMAN AHONA

Submitted By

Name	ID	Contribution
Junayed Fahad	20-43858-2	UI design,table creation,data insetation,Basic pl/sql, ,data inseartation,synonym,operators, half Querry writing, Group function, Loop,Advance PL/SQ, Explicit Cursor, Row-Level Trigger, Statement-Level Trigger
Meskat Hassan	21-45894-3	Introduction, project proposal, Introduction,Scenario description,Conclusion,Querry writing.,single row function, Procedure, Table-Based Record, Cursor-Based Record, RELATIONAL ALGEBRA
ZAHID AHAMED FAHIM	21-45866-3	ER diagram, Normalization, Schema Diagram, half Querry writing. SQL -3 view -3 synonym,joining, Group function,use case diagram,activity diagram, Package, Expextion Handling

Introduction:

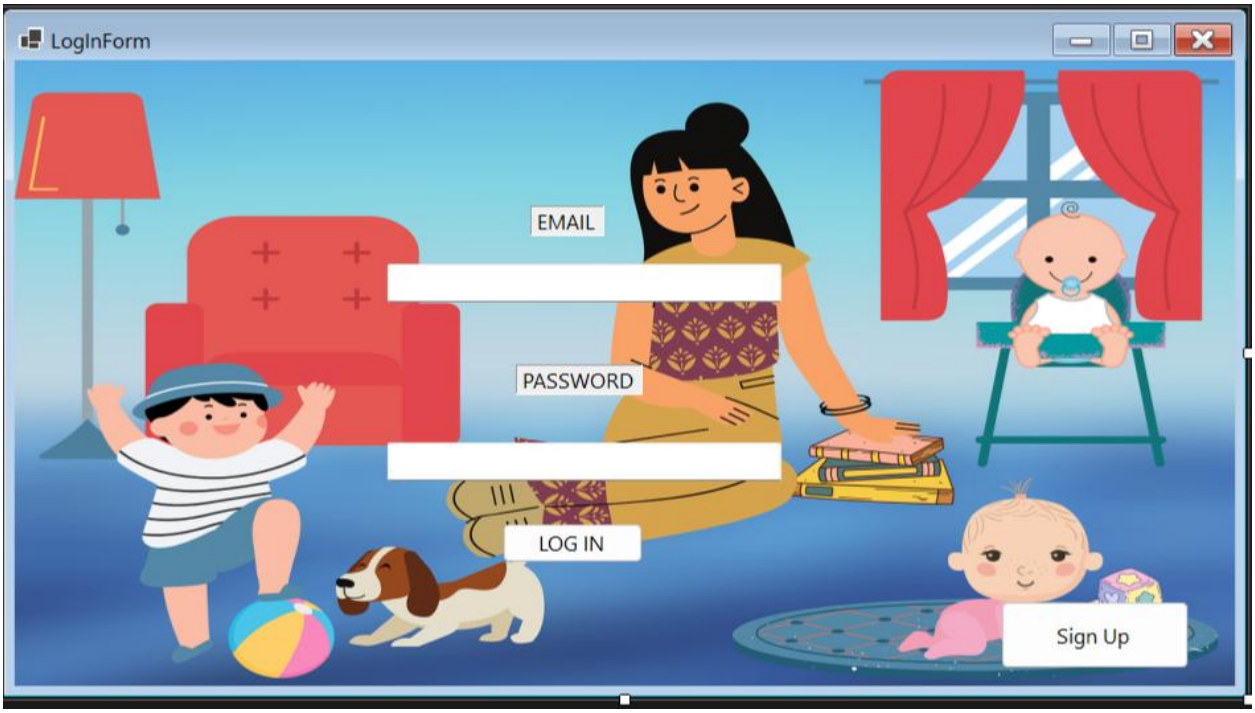
Babysitters are play a very important role in parents life whose parents are busy or their mother is working women. The Babysitter Management System is a simple and organized way to help parents to book trusted babysitters for their babys. It allows parents to register, add information about their kids, and book a babysitting services easily. Babysitters can also be added to the system, showing their availability and hourly rates. The system keeps track of bookings, payments, and feedback from parents—making the whole process smoother, safer, and more reliable for everyone involved.

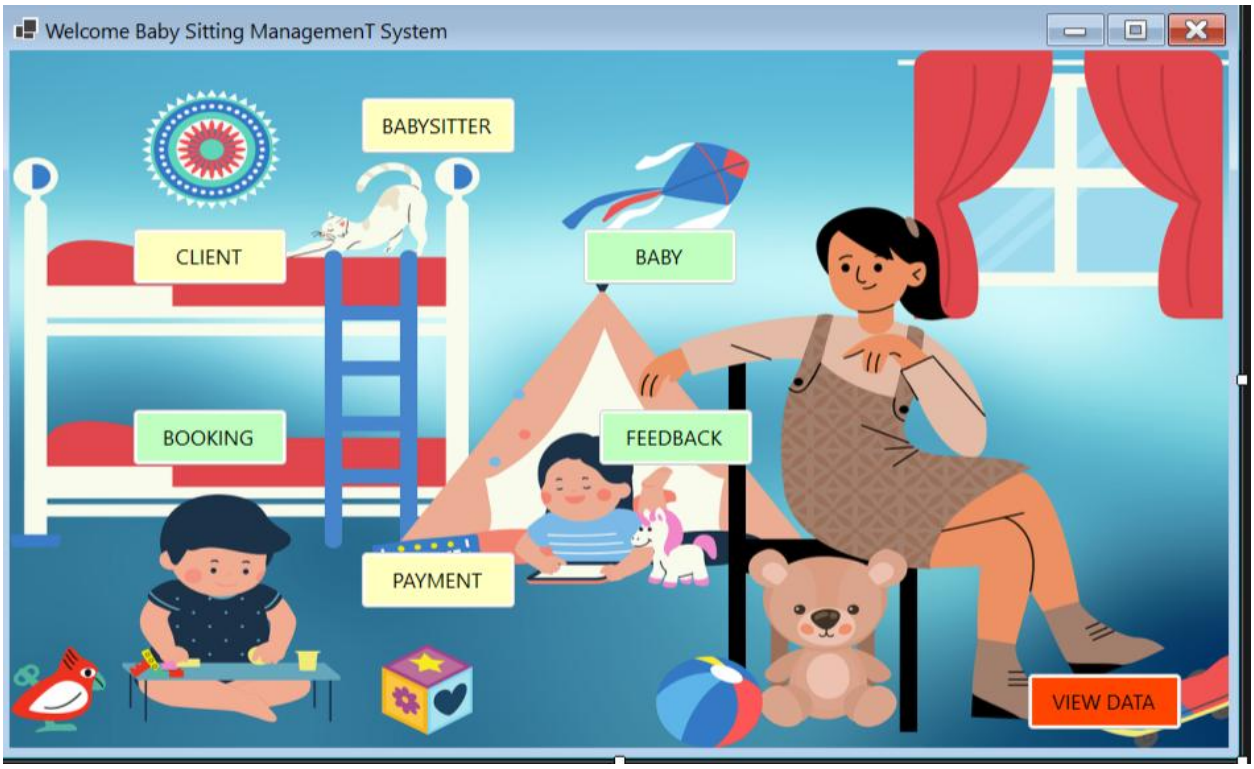
Project Proposal:

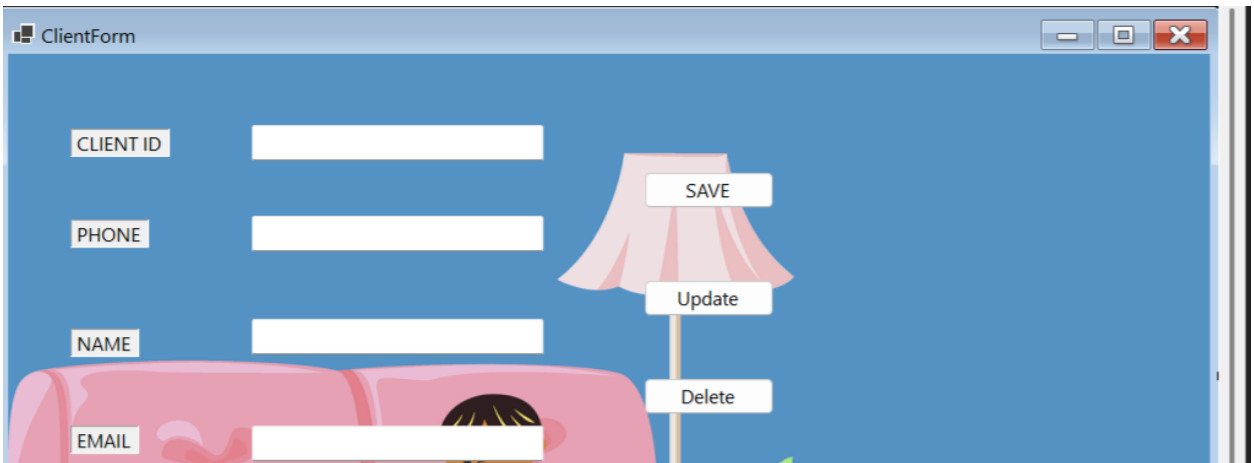
Babysitter management system is designed to help working parents and take care of there babies. Parents can register, manage their personal details, add profiles for their children, make babysitting bookings and after taking service they can give feedback about the service of babysitters Babysitters can register and can give their details like availability, contact information, and hourly ratesWhile

booking a payment record is generated. The system organize all the information securely. Manage the system seamlessly.

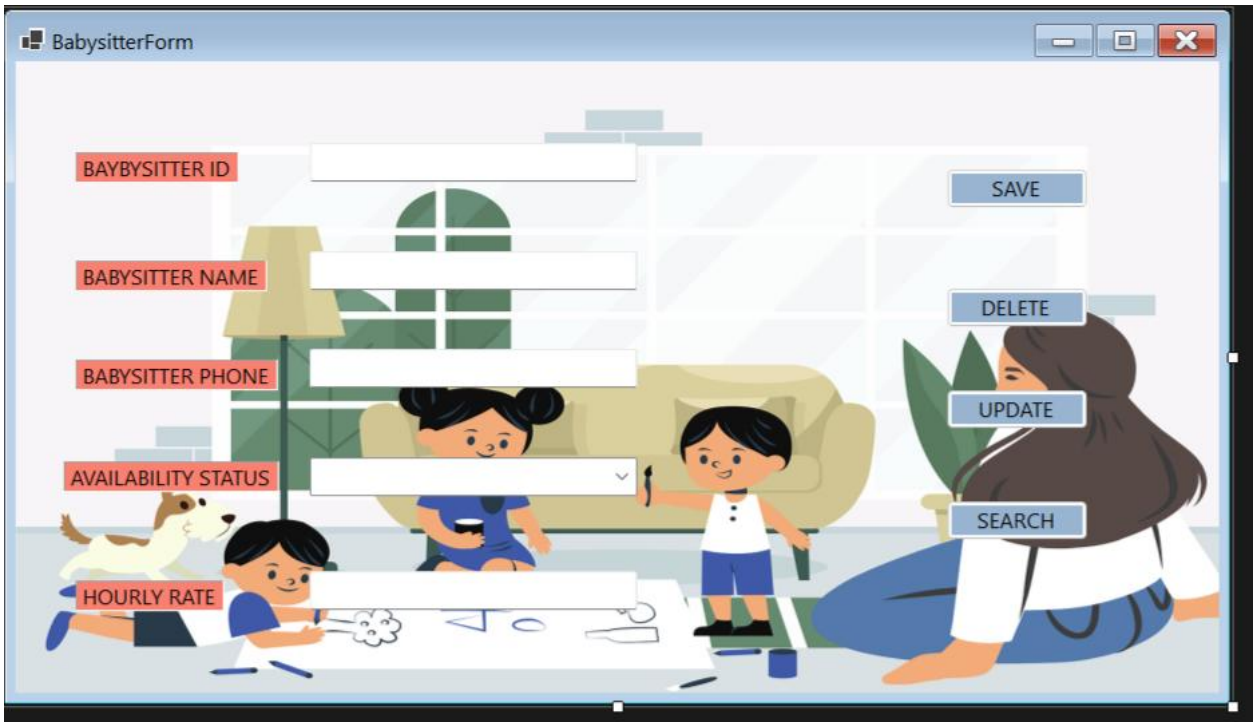
User interface:



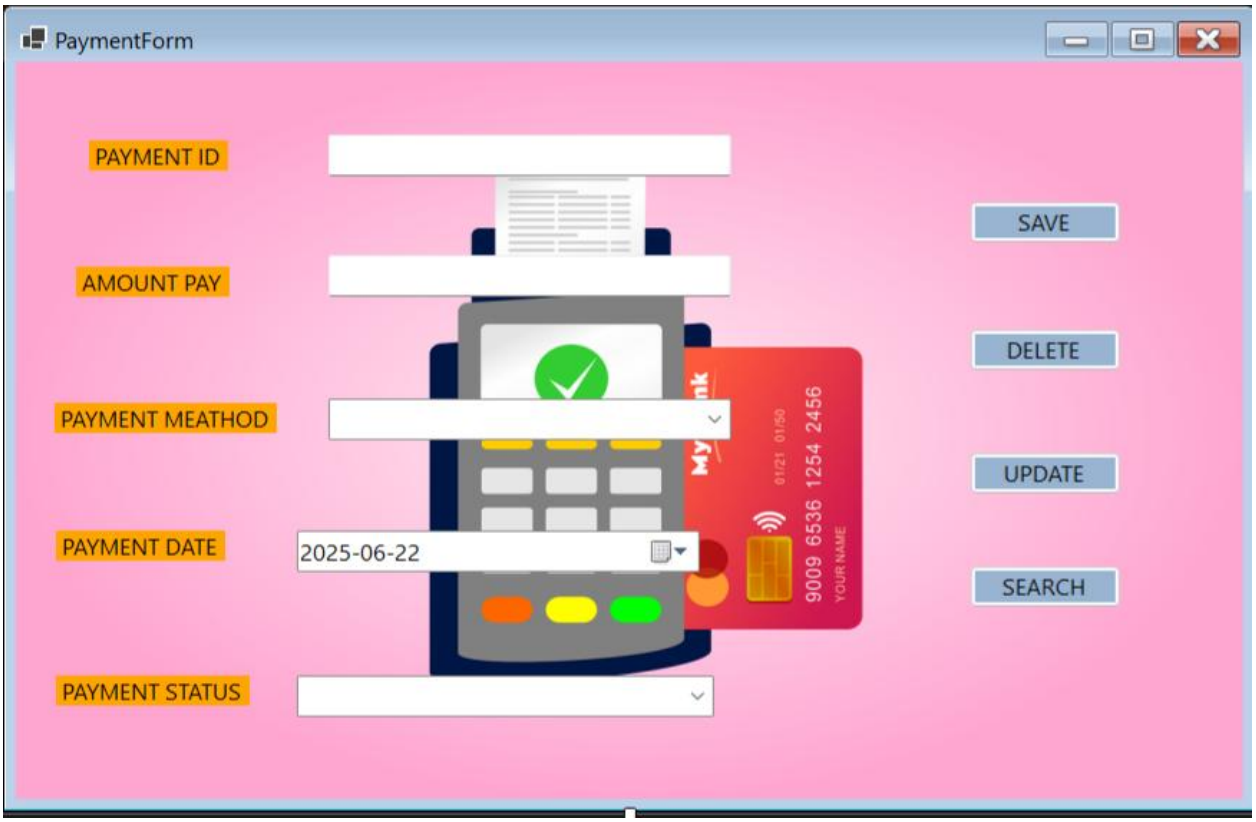

Welcome Baby Sitting ManagemenT System
BABYSITTER
CLIENT
BABY
BOOKING
FEEDBACK
PAYMENT
VIEW DATA

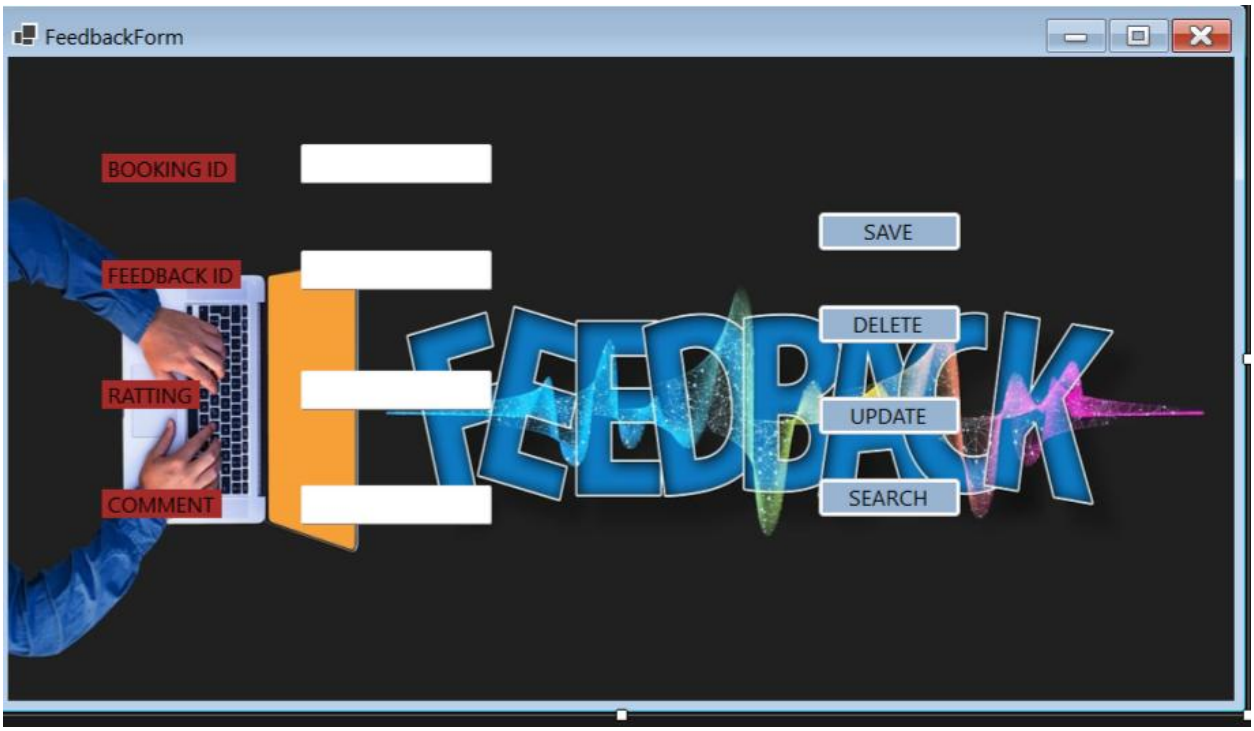

ClientForm
CLIENT ID
PHONE
NAME
EMAIL
SAVE
Update
Delete

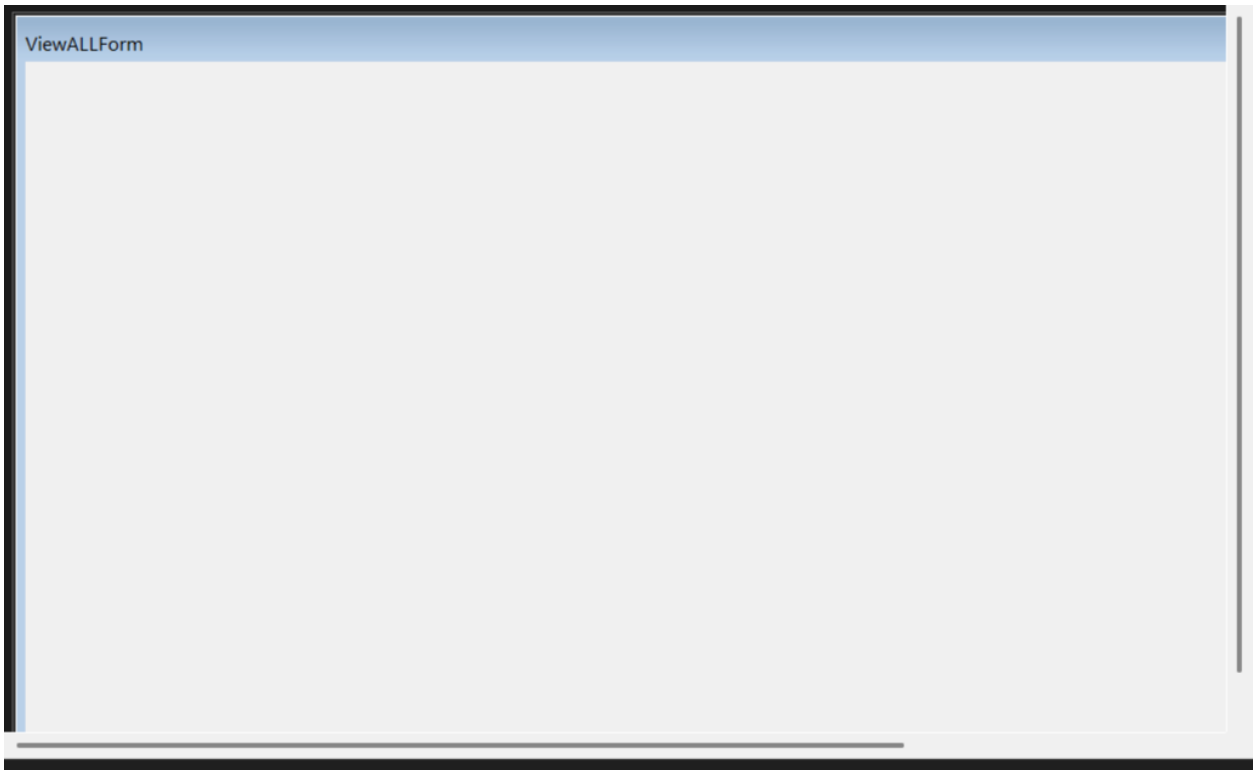

BookingForm
CLIENT ID
BOOKING ID
START TIME
END TIME
STATUS
PAYMENT ID
BABYSITTER ID
SAVE
DELETE
UPDATE
SEARCH


BabysitterForm
BAYBYSITTER ID
BABYSITTER NAME
BABYSITTER PHONE
AVAILABILITY STATUS
HOURLY RATE
SAVE
DELETE
UPDATE
SEARCH


BabyForm
CLIENT ID
BABY ID
BABY NAME
BABY AGE
BABY GENDER
BABY MEDICAL NOTE
SAVE
DELETE
SEACRH
UPDATE


PaymentForm
PAYMENT ID
AMOUNT PAY
PAYMENT MEATHOD
PAYMENT DATE
2025-06-22
PAYMENT STATUS
SAVE
DELETE
UPDATE
SEARCH


FeedbackForm
BOOKING ID
FEEDBACK ID
RATTING
COMMENT
SAVE
DELETE
UPDATE
SEARCH
FEEDBACK


ViewALLForm

Scenario Description:

In a Babysitter management system a client can have many babies. A client can have atleast one to many babies .A client can have uniquely identified by client_id, a client can have phone number, name, email, address .

A baby can have only one parents. A baby is identified by unique baby_id, also have other attributes like baby name, age, gender, medical notes.

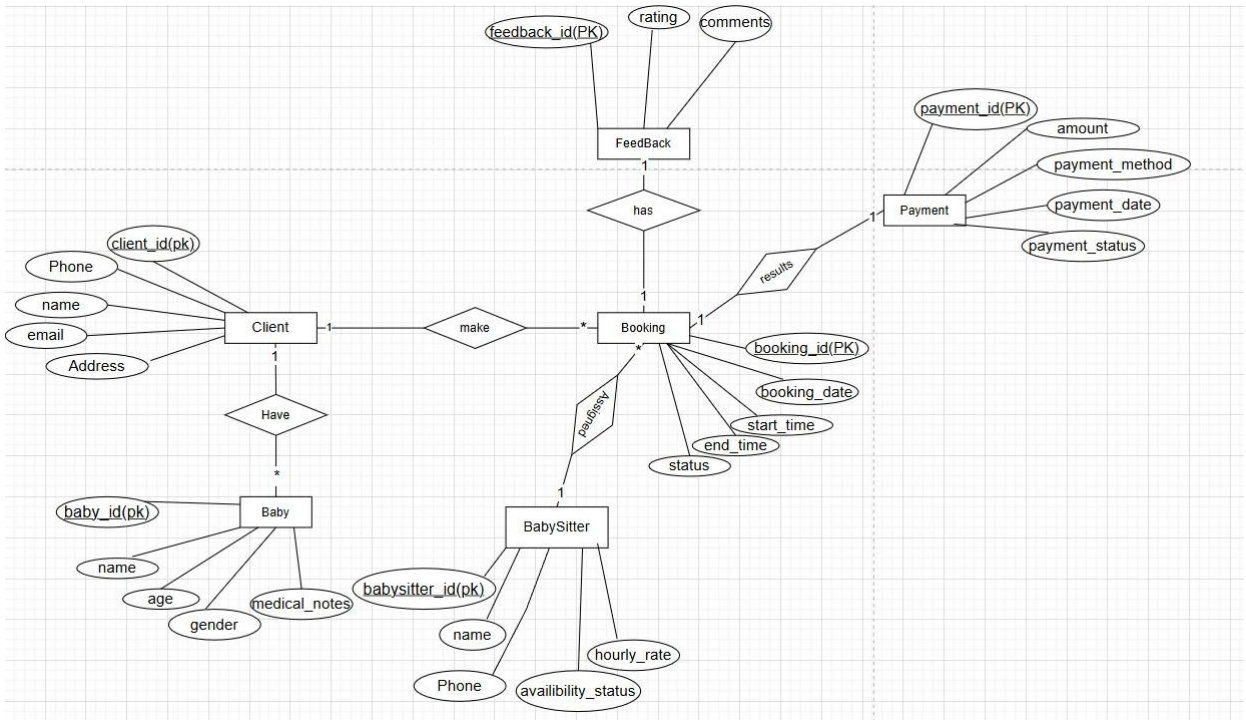
A client can make bookings, after bookings a unique booking_id is generated. A bookings can have attributes like booking_date, start time,end time, and booking status. A client can make many bookings.

Bookings are assigned to babysitter. Many bookings can assigned to a babysitter. A babysitter is identified by unique id, babysitter can have attributes name, phone, availability status, hourly rate.

A Booking results only one payment. A payment is identified by unique id, can also have attributes like payment id, amount, payment method , payment date , payment status.

A Bookings can has zero or one feedback about babysitters. A feedback is identified by unique id, rating , comments.

ER Diagram:



Normalization:

Make

<u>UNF</u>

make (**<u>client_id(pk)</u>** <u>, phone, name, email, address,</u>**<u>booking_id(pk),</u>** <u>booking_date, start_time,</u> <u>end_time, status</u>)

<u>1NF</u>

There is no multi valued attribute. Relation already in 1NF.

1. **<u>client_id(pk)</u>** <u>, phone, name, email, address,</u>**<u>booking_id(pk),</u>** <u>booking_date, start_time,</u> <u>end_time, status</u>

2NF

1. **client_id(pk)** , phone, name, email, address
2. **booking_id(pk),** booking_date, start_time, end_time, status

3NF

1. **client_id(pk)** , phone, name, email, address

2. **booking_id(pk),** booking_date, start_time, end_time, status

Table Creation

1. **client_id(pk)** , phone, name, email, address,
2. **booking_id(pk),** booking_date, start_time, end_time, status, **client_id(fk)**

Have

UNF

have (**client_id(pk)** , phone, name, email, address,**baby_id(pk),** bname, bage, bgender, bmedical_notes)

1NF

There is no multi valued attribute. Relation already in 1NF.

1. **client_id(pk)** , phone, name, email, address, **baby_id(pk),** bname, bage, bgender, bmedical_notes

2NF

1. **client_id(pk)** , phone, name, email, address
2. **baby_id(pk),** bname, bage, bgender, bmedical_notes

3NF

1. **client_id(pk)** , phone, name, email, address
2. **baby_id(pk),** bname, bage, bgender, bmedical_notes

Table Creation

1. **client_id(pk)** , phone, name, email, address, **baby_id(fk)**
2. **baby_id(pk),** bname, bage, bgender, bmedical_notes

Has

UNF

have (**booking_id(pk),** booking_date, start_time, end_time, status, **feedback_id(pk),** rating, comments)

1NF

There is no multi valued attribute. Relation already in 1NF.

1. **booking_id(pk),** booking_date, start_time, end_time, status, **feedback_id(pk),** rating, comments

2NF

1. **booking_id(pk),** booking_date, start_time, end_time, status
2. **feedback_id(pk),** rating, comments

3NF

1. **booking_id(pk),** booking_date, start_time, end_time, status
2. **feedback_id(pk),** rating, comments

Table Creation

1. **booking_id(pk),** booking_date, start_time, end_time, status
2. **feedback_id(pk),** rating, comments**, booking_id(fk)**

Results

UNF

results (**booking_id(pk),** booking_date, start_time, end_time, status, **payment_id(pk),** amount, payment_method, payment_date, payment_status)

1NF

There is no multi valued attribute. Relation already in 1NF.

1. **booking_id(pk),** booking_date, start_time, end_time, status, **payment_id(pk),** amount, payment_method, payment_date, payment_status

2NF

1. **booking_id(pk),** booking_date, start_time, end_time, status
2. **payment_id(pk),** amount, payment_method, payment_date, payment_status

3NF

1. **booking_id(pk),** booking_date, start_time, end_time, status
2. **payment_id(pk),** amount, payment_method, payment_date, payment_status

Table Creation

1. **booking_id(PK),** booking_date, start_time, end_time, status, **payment_id(FK)**
2. **payment_id(PK),** amount, payment_method, payment_date, payment_status

Assigned

UNF

results (**booking_id(PK),** booking_date, start_time, end_time, status, **babysitter_id(PK),** bsname, bsphone, availibility_status, hourly_rate)

1NF

There is no multi valued attribute. Relation already in 1NF.

2. **booking_id(PK),** booking_date, start_time, end_time, status, **babysitter_id(PK),** bsname, bsphone, vailability_status, hourly_rate

2NF

1. **booking_id(PK),** booking_date, start_time, end_time, status
2. **babysitter_id(PK),** bsname, bsphone, vailability_status, hourly_rate

3NF

1. **booking_id(PK),** booking_date, start_time, end_time, status
2. **babysitter_id(PK),** bsname, bsphone, vailability_status, hourly_rate

Table Creation:

1. **booking_id(PK),** booking_date, start_time, end_time, status , **babysitter_id(FK)**
2. **babysitter_id(PK),** bsname, bsphone, vailability_status, hourly_rate

Temporary Tables:

1) **client_id(pk)** , phone, name, email, address
2) **booking_id(pk),** booking_date, start_time, end_time, status, **client_id(fk)**
3) ~~**client_id(pk)** , phone, name, email, address~~
4) **baby_id(pk),** bname, bage, bgender, bmedical_notes, **client_id(FK)**
5) ~~**booking_id(pk),** booking_date, start_time, end_time, status~~
6) **feedback_id(pk),** rating, comments**, booking_id(fk)**
7) **booking_id(PK),** booking_date, start_time, end_time, status, **payment_id(FK)**

8) **payment_id(PK),** amount, payment_method, payment_date, payment_status
9) **booking_id(PK),** booking_date, start_time, end_time, status , **babysitter_id(FK)** 10) **babysitter_id(PK),** bsname, bsphone, vailability_status, hourly_rate

Final Tables:

1) **client_id(pk)** , phone, name, email, address
2) **booking_id(pk),** booking_date, start_time, end_time, status, **client_id(fk), payment_id(FK) , babysitter_id(FK)**

3) **baby_id(pk),** bname, bage, bgender, bmedical_notes, **client_id(FK)**

4) **feedback_id(pk),** rating, comments**, booking_id(fk)**
5) **payment_id(PK),** amount, payment_method, payment_date, payment_status
6) **babysitter_id(PK),** bsname, bsphone, vailability_status, hourly_rate

Schema Diagram:

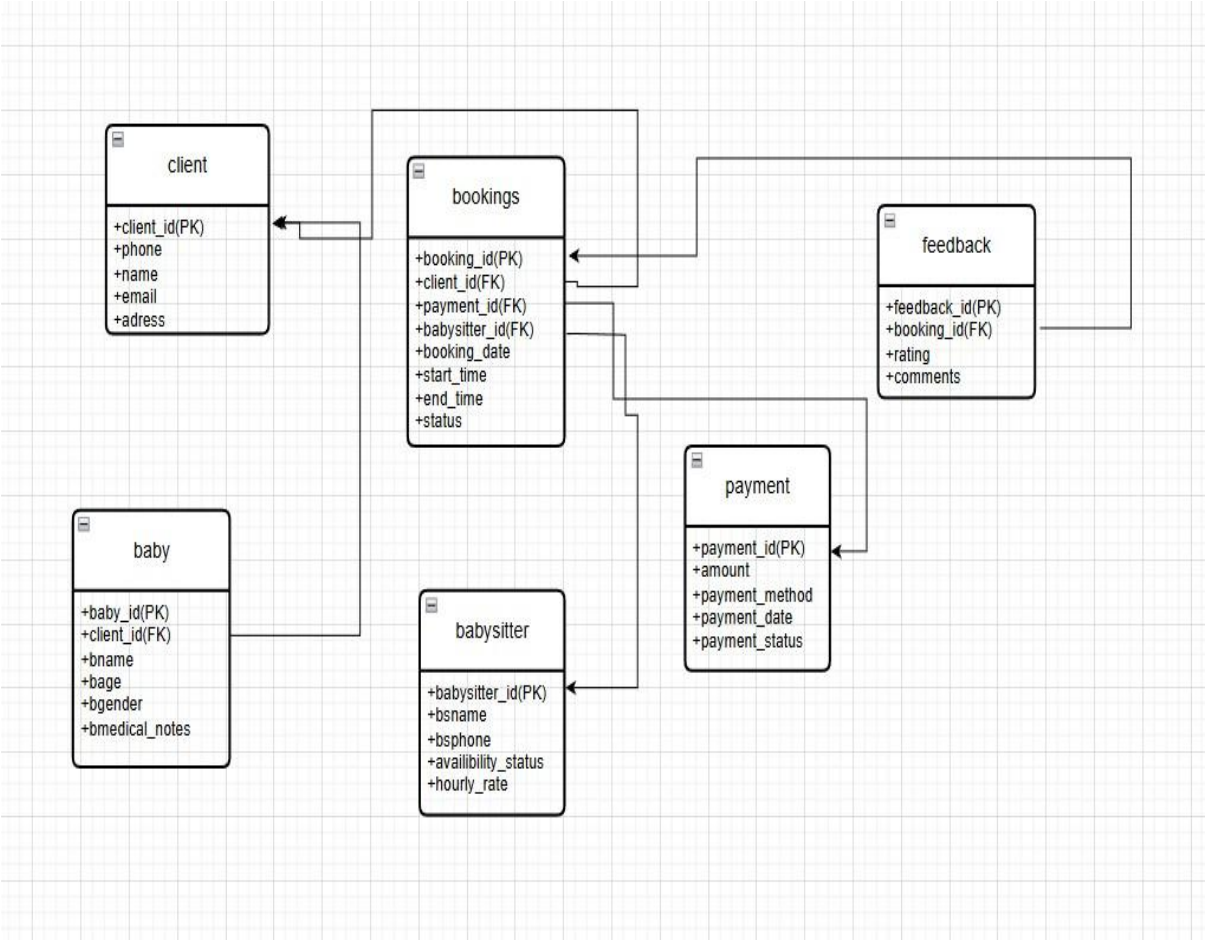


Table Creation

CLIENT TABLE:

CREATE TABLE CLIENT (

CLIENT_ID VARCHAR2(255) PRIMARY KEY,

Phone VARCHAR2(50),

Name VARCHAR2(255),

Email VARCHAR2(255), "For drope this table you net to write this query//DROP TABLE CLIENT CASCADE CONSTRAINTS;"

Address VARCHAR2(255)

);

```
CREATE TABLE CLIENT (
    CLIENT_ID VARCHAR2(255) PRIMARY KEY,
    Phone VARCHAR2(50),
    Name VARCHAR2(255),
    Email VARCHAR2(255),

    Address VARCHAR2(255)
);
CREATE INDEX idx_client_name ON Client(name);
```

Results Explain Describe Saved SQL History

```
ORA-00955: name is already used by an existing object
```

BABYSITTER TABLE:

```
CREATE TABLE Babysitter (
    Babysitter_ID VARCHAR(255) PRIMARY KEY,
    BabySName VARCHAR(255),
    BabySPhone VARCHAR(50),
    Availability_Status VARCHAR(100),
    Hourly_Rate NUMBER(6,2)
);
```

PAYMENR TABLE:

```
CREATE TABLE Payment (
    Payment_ID VARCHAR(10) PRIMARY KEY,
    Amount NUMBER(10,2),
    Payment_Method VARCHAR(50),
    Payment_Date DATE,
    Payment_Status VARCHAR(255)
);
```

BOOKING TABLE:

```
CREATE TABLE Booking (
    Booking_ID INT PRIMARY KEY,
    Booking_Date DATE,
    Start_Time VARCHAR(20),
    End_Time VARCHAR(20),
    Status VARCHAR(50),
    Client_ID VARCHAR(10),
    Payment_ID VARCHAR(10),
    Babysitter_ID VARCHAR(10),
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID),
    FOREIGN KEY (Payment_ID) REFERENCES Payment(Payment_ID),
    FOREIGN KEY (Babysitter_ID) REFERENCES Babysitter(Babysitter_ID)
);
```

Autocommit Display 10

```
CREATE TABLE Booking (
    Booking_ID INT PRIMARY KEY,
    Booking_Date DATE,
    Start_Time VARCHAR(20),
    End_Time VARCHAR(20),
    Status VARCHAR(50),
    Client_ID VARCHAR(10),
    Payment_ID VARCHAR(10),
    Babysitter_ID VARCHAR(10),
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID),
    FOREIGN KEY (Payment_ID) REFERENCES Payment(Payment_ID),
    FOREIGN KEY (Babysitter_ID) REFERENCES Babysitter(Babysitter_ID)
);

CRE  NDEX idx_booking_status ON Booking(Status);

-- Create Sequence
CREATE SEQUENCE seq_booking START WITH 1 INCREMENT BY 1;
```

Results Explain Describe Saved SQL History

```
ORA-00955: name is already used by an existing object
```

BABY TABLE:

```
CREATE TABLE Baby (
    Baby_ID INT PRIMARY KEY,
    BabyName VARCHAR(255),
    BabyAge INT,
    BabyGender VARCHAR(50),
    BabyMedical_Notes VARCHAR(500),
    Client_ID VARCHAR(10),
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID)
);
```

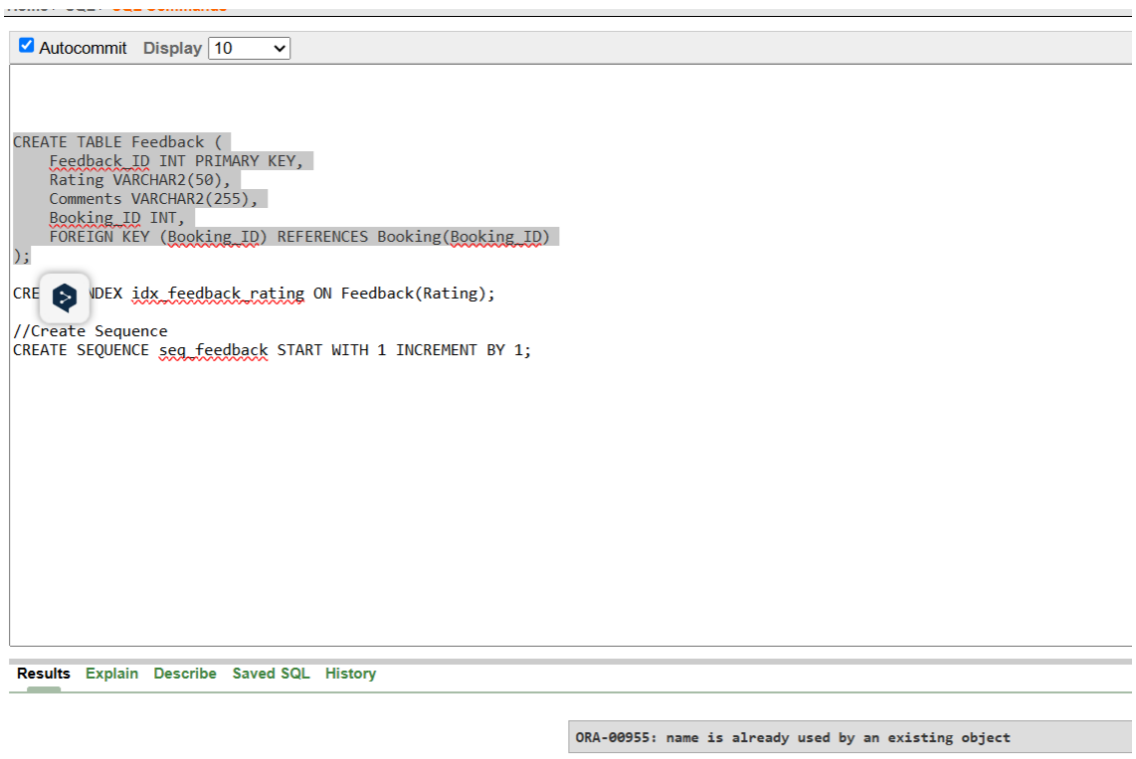
```
CREATE TABLE Baby (
    Baby_ID INT PRIMARY KEY,
    BabyName VARCHAR(255),
    BabyAge INT,
    BabyGender VARCHAR(50),
    BabyMedical_Notes VARCHAR(500),
    Client_ID VARCHAR(10),
    FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID)
);

CREATE INDEX idx_baby_name ON Baby(BabyName);

//Create Sequence
CREATE SEQUENCE seq_baby START WITH 1 INCREMENT BY 1;

CREATE TABLE Feedback (
```

Results Explain Describe Saved SQL History

```
ORA-00955: name is already used by an existing object
```

FEEDBACK TABLE:

```
CREATE TABLE Feedback (

   Feedback_ID INT PRIMARY KEY,

   Rating VARCHAR2(50),

   Comments VARCHAR2(255),

   Booking_ID INT,

   FOREIGN KEY (Booking_ID) REFERENCES Booking(Booking_ID)

);
```



Create Sequence:

```
CREATE SEQUENCE seq_booking START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq_baby START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE seq_feedback START WITH 1 INCREMENT BY 1;
```

Create Index

```
--Create Index

CREATE INDEX idx_client_name ON Client(name);

CREATE INDEX idx_bsname ON Babysitter(BabySName);

CREATE INDEX idx_payment_method ON Payment(Payment_Method);

CREATE INDEX idx_booking_status ON Booking(Status);

CREATE INDEX idx_baby_name ON Baby(BabyName);

CREATE INDEX idx_feedback_rating ON Feedback(Rating);
```

DATA INSERTATION:

CLIENT DATA:

```
INSERT INTO CLIENT VALUES ('1', '555', 'Smith', 'smith@', 'Ek');

INSERT INTO CLIENT VALUES ('2', '555', 'Smith', 'smith@', 'Ek');


INSERT INTO CLIENT VALUES ('4', '555', 'mith', 'mith@', 'Ek');

INSERT INTO CLIENT VALUES ('5', '555', 'Smith', 'smith@', 'Ek');
```

BABYSITTER DATA:

INSERT INTO Babysitter VALUES ('1', 'Alice', '90', 'yes', 15.00);

INSERT INTO Babysitter VALUES ('3', 'Alice', '90', 'yes', 15.00);

INSERT INTO Babysitter VALUES ('4', 'Alice', '90', 'yes', 15.00);

INSERT INTO Babysitter VALUES ('4', 'Alice', '90', 'yes', 15.00);

PAYMENT DATA:

INSERT INTO Payment VALUES ('1', 100.00, 'credit', '2024-12-01', 'complete');

INSERT INTO Payment VALUES ('2', 100.00, 'credit', '2024-12-01', 'complete');

INSERT INTO Payment VALUES ('3', 100.00, 'credit', '2024-12-01', 'complete');

INSERT INTO Payment VALUES ('4', 100.00, 'credit', '2024-12-01', 'complete');

BOOKING DATA:

INSERT INTO Booking VALUES (1, '2025-05-15', '10', '12', 'YES', '1', '1', '1');

INSERT INTO Booking VALUES (2, '2025-05-15', '10', '12', 'YES', '1', '1', '1');

INSERT INTO Booking VALUES (3, '2025-05-15', '10', '12', 'YES', '1', '1', '1');

INSERT INTO Booking VALUES (4, '2025-05-15', '10', '12', 'YES', '1', '1', '1');

BABY DATA:

INSERT INTO Baby VALUES ('1', 'Liam', 2, 'Male', 'allergies', '1');

INSERT INTO Baby VALUES ('2', 'Liam', 2, 'Male', 'allergies', '2');

INSERT INTO Baby VALUES ('3', 'Liam', 2, 'Male', 'allergies', '1');

INSERT INTO Baby VALUES ('4’, 'Liam', 2, 'Male', 'allergies', '2');

FEEDBACK DATA:

INSERT INTO FEEDBACK VALUES(‘1’,’1’,’5star’,’good’);

INSERT INTO FEEDBACK VALUES(‘2’,’2’,’5star’,’good’);

INSERT INTO FEEDBACK VALUES(‘2’,’2’,’5star’,’good’);

INSERT INTO FEEDBACK VALUES(‘2’,’2’,’5star’,’good’);

Query Writing

SQL

VIEW

```
CREATE VIEW Client_details AS
SELECT
   CLIENT_ID,Name,Phone,Email,Address
FROM
   Client
WHERE
```

Email IS NOT NULL;

Autocommit Display 10

```
CREATE VIEW Client_details AS
SELECT
    CLIENT_ID,Name,Phone,Email,Address
FROM
    Client
WHERE
    Email IS NOT NULL;


desc Client_details
```

Results Explain Describe Saved SQL History

Object Type VIEW Object CLIENT_DETAILS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CLIENT_DETAILS	CLIENT_ID	Varchar2	255	-	-	-	-	-	-
	NAME	Varchar2	255	-	-	-	✓	-	-
	PHONE	Varchar2	50	-	-	-	✓	-	-
	EMAIL	Varchar2	255	-	-	-	✓	-	-
	ADDRESS	Varchar2	255	-	-	-	✓	-	-

1 - 5

Create views for Babysitter:

CREATE VIEW Babysitter_details AS

SELECT Babysitter_ID, BabySName, BabySPhone, Availability_Status, Hourly_Rate

FROM Babysitter

WHERE Availability_Status = 'YES'

AND Hourly_Rate >= 100;

Home > SQL > SQL Commands

Autocommit Display 10

```
CREATE VIEW Babysitter_details AS

SELECT Babysitter_ID, BabySName, BabySPhone, Availability_Status, Hourly_Rate

FROM Babysitter

WHERE Availability_Status = 'YES'

  AND Hourly_Rate >= 100;
```

Results Explain Describe Saved SQL History

```
View created.
```

0.00 seconds

```
desc babysitter_details
```

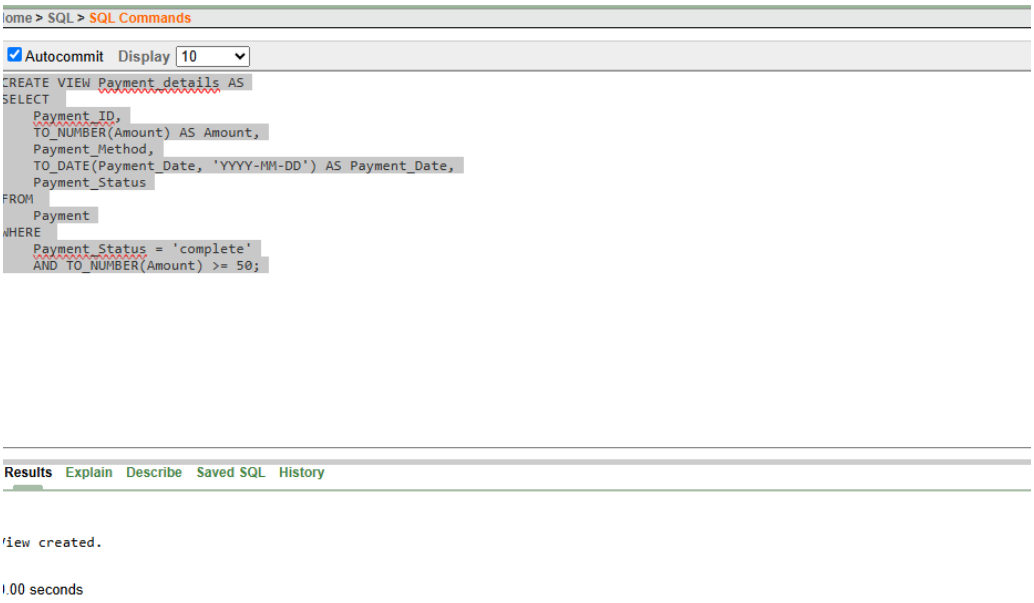
Results Explain Describe Saved SQL History

Object Type VIEW Object BABYSITTER_DETAILS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BABYSITTER_DETAILS	BABYSITTER_ID	Varchar2	255	-	-	-	-	-	-
	BABYSNAME	Varchar2	255	-	-	-	✓	-	-
	BABYSPHONE	Varchar2	50	-	-	-	✓	-	-
	AVAILABILITY_STATUS	Varchar2	100	-	-	-	✓	-	-
	HOURLY_RATE	Number	-	10	2	-	✓	-	-
									1 - 5

Create views For Payment:

```
CREATE VIEW Payment_details AS
SELECT
    Payment_ID,
    TO_NUMBER(Amount) AS Amount,
    Payment_Method,
    TO_DATE(Payment_Date, 'YYYY-MM-DD') AS Payment_Date,
    Payment_Status
FROM    Payment WHERE
    Payment_Status = 'complete'
    AND TO_NUMBER(Amount) >= 50;
```



Autocommit Display 10

```
CREATE VIEW Payment_details AS
SELECT
    Payment_ID,
    TO_NUMBER(Amount) AS Amount,
    Payment_Method,
    TO_DATE(Payment_Date, 'YYYY-MM-DD') AS Payment_Date,
    Payment_Status
FROM
    Payment
WHERE
    Payment_Status = 'complete'
    AND TO_NUMBER(Amount) >= 50;


desc Payment_details
```

Results Explain Describe Saved SQL History

Object Type **VIEW** Object **PAYMENT_DETAILS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT_DETAILS	PAYMENT_ID	Varchar2	255	-	-	-	-	-	-
	AMOUNT	Number	-	-	-	-	✓	-	-
	PAYMENT_METHOD	Varchar2	255	-	-	-	✓	-	-
	PAYMENT_DATE	Date	8	-	-	-	✓	-	-
	PAYMENT_STATUS	Varchar2	255	-	-	-	✓	-	-

1 - 5

Create views For Booking:

CREATE VIEW Booking_details AS

SELECT

Booking_ID,Booking_Date,Start_Time,End_Time,Status,Client_ID,Payment_ID,Babysitter_ID

FROM

Booking

WHERE

Status = 'complete';

Autocommit Display 10

```
CREATE VIEW Booking_details AS
SELECT
    Booking_ID,Booking_Date,Start_Time,End_Time,Status,Client_ID,Payment_ID,Babysitter_ID
FROM
    Booking
WHERE
    Status = 'complete';
```

Results Explain Describe Saved SQL History

```
ORA-00955: name is already used by an existing object
```

Home > SQL > SQL Commands

Autocommit Display 10

```
CREATE VIEW Booking_details AS
SELECT
    Booking_ID,Booking_Date,Start_Time,End_Time,Status,Client_ID,Payment_ID,Babysitter_ID
FROM
    Booking
WHERE
    Status = 'complete';
```

```
desc booking_details
```

Results Explain Describe Saved SQL History

Object Type **VIEW** Object **BOOKING_DETAILS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKING_DETAILS	BOOKING_ID	Varchar2	50	-	-	-	-	-	-
	BOOKING_DATE	Varchar2	50	-	-	-	-	-	-
	START_TIME	Varchar2	20	-	-	-	✓	-	-
	END_TIME	Varchar2	20	-	-	-	✓	-	-
	STATUS	Varchar2	255	-	-	-	-	-	-
	CLIENT_ID	Varchar2	255	-	-	-	-	-	-
	PAYMENT_ID	Varchar2	10	-	-	-	✓	-	-
	BABYSITTER_ID	Varchar2	10	-	-	-	✓	-	-

1 - 8

SYNONYM

Create synonym for Client table:

create synonym cli_cli for client;

```
create synonym cli_cli for client;
```

Results Explain Describe Saved SQL History

```
Synonym created.
```

0.00 seconds

```
create synonym cli_cli for client;

select * from cli_cli
```

Results Explain Describe Saved SQL History

CLIENT_ID	PHONE	NAME	EMAIL	ADDRESS
1	13	ju@	ju	du
2	013	junayed@	Junayed	khulna

2 rows returned in 0.00 seconds CSV Export

Create synonym for Babysitter:

create synonym bbs_bbs for babysitter;

```
create synonym bbs_bbs for babysitter;
```

Results Explain Describe Saved SQL History

```
Synonym created.
```

0.00 seconds

```
select * from bbs_bbs
```

Results Explain Describe Saved SQL History

BABYSITTER_ID	BABYSNAME	BABYSPHONE	AVAILABILITY_STATUS	HOURLY_RATE
1	as	19	YES	12.3
2	saima	018	YES	110

2 rows returned in 0.00 seconds CSV Export

Create synonym for Baby:

create synonym bby_bby for baby;

```
create synonym bby_bby for baby;
```

Results Explain Describe Saved SQL History

```
Synonym created.
```

0.00 seconds

```
select * from bby_bby
```

Results Explain Describe Saved SQL History

BABY_ID	BABYNAME	BABYAGE	BABYGENDER	BABYMEDICAL_NOTES	CLIENT_ID
1	sima	4	female	pain	1
2	Isa	12	FEMALE	Dust Allergi	2

2 rows returned in 0.00 seconds CSV Export

Create synonym for Booking:

create synonym book_book for booking;

```
create synonym book_book for booking;
```

Results Explain Describe Saved SQL History

```
Synonym created.
```

0.01 seconds

Results Explain Describe Saved SQL History

BOOKING_ID	BOOKING_DATE	START_TIME	END_TIME	STATUS	CLIENT_ID	PAYMENT_ID	BABYSITTER_ID
1	11-MAY-25 02.04.41.465595200 PM	1	12	complete	1	1	1
2	12-MAY-25 12.00.00.000000000 AM	10	12	complete	2	2	2
26	2025-05-15	09:00 AM	12:00 PM	Confirmed	1	1	1

3 rows returned in 0.00 seconds CSV Export

Create synonym for feedback:

create synonym fdb_fdb for feedback;

```
create synonym fdb_fdb for feedback;
```

Results Explain Describe Saved SQL History

```
Synonym created.
```

0.00 seconds

```
select * from fdb_fdb
```

Results Explain Describe Saved SQL History

FEEDBACK_ID	RATING	COMMENTS	BOOKING_ID
21	Good	gg	1
1	5star	satisfied	1
2	3star	average	2

3 rows returned in 0.00 seconds CSV Export

Basic PL/SQL

1.Variables:

For client:

```
DECLARE
    v_client_name CLIENT.Name%TYPE;
```

BEGIN

SELECT Name

INTO v_client_name

FROM CLIENT

WHERE CLIENT_ID = '1'; -- Replace '1' with a valid CLIENT_ID in your table

DBMS_OUTPUT.PUT_LINE('Client Name: ' || v_client_name);

END;

/

```
DECLARE
    v_client_name CLIENT.Name%TYPE;
BEGIN

    SELECT Name
    INTO v_client_name
    FROM CLIENT
    WHERE CLIENT_ID = '1';  -- Replace '1' with a valid CLIENT_ID in your table

    DBMS_OUTPUT.PUT_LINE('Client Name: ' || v_client_name);
END;
/
```

```
DECLARE
```

Results Explain Describe Saved SQL History

```
Client Name: ju@

Statement processed.

0.00 seconds
```

For Babysitter:

```
DECLARE
   v_babysitter_name Babysitter.BabySName%TYPE;
BEGIN

   SELECT BabySName
   INTO v_babysitter_name
   FROM Babysitter
   WHERE Babysitter_ID = '1';  -- Replace '1' with a valid
Babysitter_ID



   DBMS_OUTPUT.PUT_LINE('Babysitter Name: ' ||
v_babysitter_name);
END;
/
```

```
DECLARE
    v_babysitter_name Babysitter.BabySName%TYPE;
BEGIN

    SELECT BabySName
    INTO v_babysitter_name
    FROM Babysitter
    WHERE Babysitter_ID = '1';  -- Replace '1' with a valid Babysitter_ID


    DBMS_OUTPUT.PUT_LINE('Babysitter Name: ' || v_babysitter_name);
END;
/
```

Results Explain Describe Saved SQL History

```
Babysitter Name: as

Statement processed.

0.00 seconds
```

For Booking:

```
DECLARE
   v_booking_id Booking.Booking_ID%TYPE;
BEGIN


   SELECT Booking_ID
```

INTO v_booking_id

FROM Booking

WHERE Booking_ID = 1; -- Replace '1' with a valid Booking_ID

DBMS_OUTPUT.PUT_LINE('Booking ID: ' || v_booking_id);

END;

/

```
DECLARE
    v_booking_id Booking.Booking_ID%TYPE;
BEGIN

    SELECT Booking_ID
    INTO v_booking_id
    FROM Booking
    WHERE Booking_ID = 1;  -- Replace '1' with a valid Booking_ID


    DBMS_OUTPUT.PUT_LINE('Booking ID: ' || v_booking_id);
END;
/
```

Results Explain Describe Saved SQL History

```
Booking ID: 1

Statement processed.
```

0.00 seconds

Single-row function

```
-- Convert babysitter name to upper case

DECLARE

   v_name Babysitter.BabySName%TYPE;

BEGIN

   SELECT UPPER(BabySName) INTO v_name FROM Babysitter WHERE Babysitter_ID = '1';

   DBMS_OUTPUT.PUT_LINE('Babysitter Name in UPPERCASE: ' || v_name);

END;

/
```

```
-- Convert babysitter name to upper case
DECLARE
    v_name Babysitter.BabySName%TYPE;
BEGIN
    SELECT UPPER(BabySName) INTO v_name FROM Babysitter WHERE Babysitter_ID = '1';
    DBMS_OUTPUT.PUT_LINE('Babysitter Name in UPPERCASE: ' || v_name);
END;
/
```

Results Explain Describe Saved SQL History

```
Babysitter Name in UPPERCASE: ALICE

Statement processed.
```

0.00 seconds

Operators:

****See all the total payment:**

```
DECLARE
   v_amount Payment.Amount%TYPE;
   v_tax NUMBER;
   v_total_amount NUMBER;
BEGIN
   -- Get the Payment Amount for a specific Booking ID
   SELECT p.Amount
   INTO v_amount
   FROM Booking b
   JOIN Payment p ON b.Payment_ID = p.Payment_ID
   WHERE b.Booking_ID = 1;

   -- Calculate 5% tax
   v_tax := v_amount * 0.05;

   -- Calculate total amount with tax
   v_total_amount := v_amount + v_tax;

   -- Output result
   DBMS_OUTPUT.PUT_LINE('Total Payment (with 5% Tax): ' || v_total_amount);
END;
```

```
DECLARE
    v_amount Payment.Amount%TYPE;
    v_tax NUMBER;
    v_total_amount NUMBER;
BEGIN
    -- Get the Payment Amount for a specific Booking ID
    SELECT p.Amount
    INTO v_amount
    FROM Booking b
    JOIN Payment p ON b.Payment_ID = p.Payment_ID
    WHERE b.Booking_ID = 1;

    -- Calculate 5% tax
    v_tax := v_amount * 0.05;
```

Results Explain Describe Saved SQL History

```
Total Payment (with 5% Tax): 12.6

Statement processed.
```

0.00 seconds

Check Payment Status:

See the payment is complete or not :

```
DECLARE
   v_status Payment.Payment_Status%TYPE;
   v_amount Payment.Amount%TYPE;
BEGIN
   SELECT Payment_Status, Amount
   INTO v_status, v_amount
   FROM Payment
   WHERE Payment_ID = 'P001'; -- Change as needed
```

```
    IF UPPER(v_status) = 'COMPLETE' THEN
        DBMS_OUTPUT.PUT_LINE('Payment is complete. Amount: ' || v_amount);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Payment is not complete. Status: ' || v_status);
    END IF;
END;
/
```

Autocommit Display 10 Save

```
DECLARE
    v_status Payment.Payment_Status%TYPE;
    v_amount Payment.Amount%TYPE;
BEGIN
    SELECT Payment_Status, Amount
    INTO v_status, v_amount
    FROM Payment
    WHERE Payment_ID = '1'; -- Change as needed

    IF UPPER(v_status) = 'COMPLETE' THEN
        DBMS_OUTPUT.PUT_LINE('Payment is complete. Amount: ' || v_amount);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Payment is not complete. Status: ' || v_status);
    END IF;
```

Results Explain Describe Saved SQL History

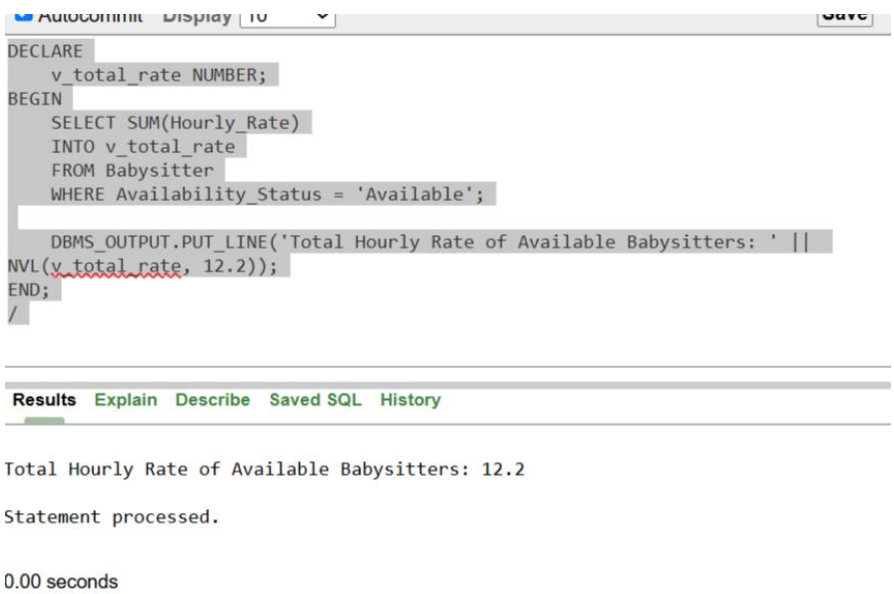
```
Payment is complete. Amount: 12

Statement processed.
```

0.00 seconds

Group function:

**See the total hourly_rate of the babysitter

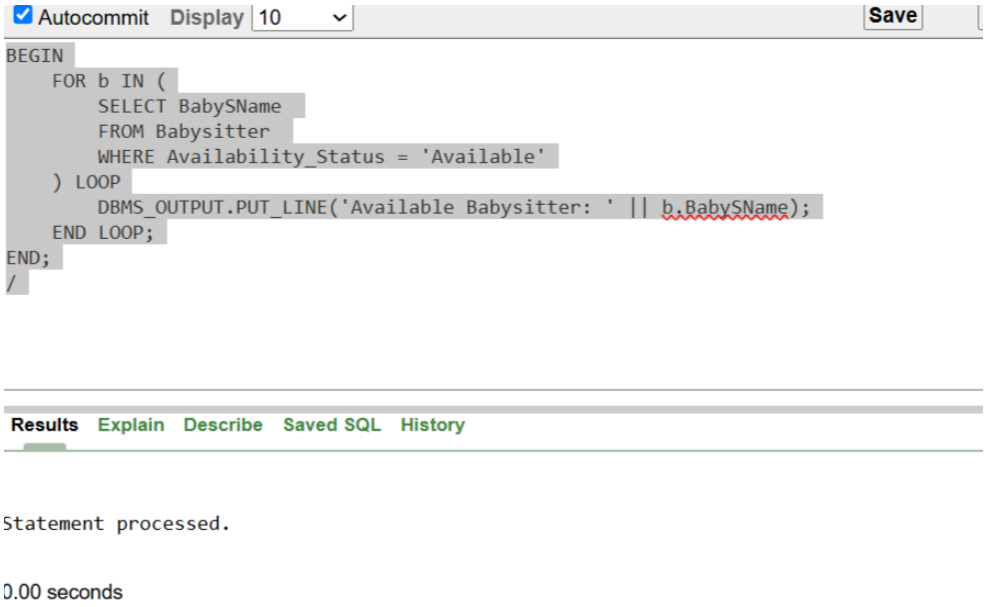
```
DECLARE
   v_total_rate NUMBER;
BEGIN
   SELECT SUM(Hourly_Rate)
   INTO v_total_rate
   FROM Babysitter
   WHERE Availability_Status = 'Available';

   DBMS_OUTPUT.PUT_LINE('Total Hourly Rate of Available Babysitters: ' || NVL(v_total_rate, 0));
END;
/
```



Loop:

Babysitter is available or not:

```
BEGIN
   FOR b IN (
      SELECT BabySName
      FROM Babysitter
      WHERE Availability_Status = 'Available'
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Available Babysitter: ' || b.BabySName);
   END LOOP;
END;
/
```



**Payment is more than 100 or less than 100

Payment Table (Amount < 100):

```
BEGIN
   FOR p IN (
      SELECT Payment_ID, Amount
      FROM Payment
      WHERE Amount < 100
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Payment ID ' || p.Payment_ID || ' amount: ' || p.Amount);
   END LOOP;
END;
/
```

```
BEGIN
    FOR p IN (
        SELECT Payment_ID, Amount
        FROM Payment
        WHERE Amount < 100
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Payment ID ' || p.Payment_ID || ' amount: ' || p.Amount);
    END LOOP;
END;
/
```

Results Explain Describe Saved SQL History

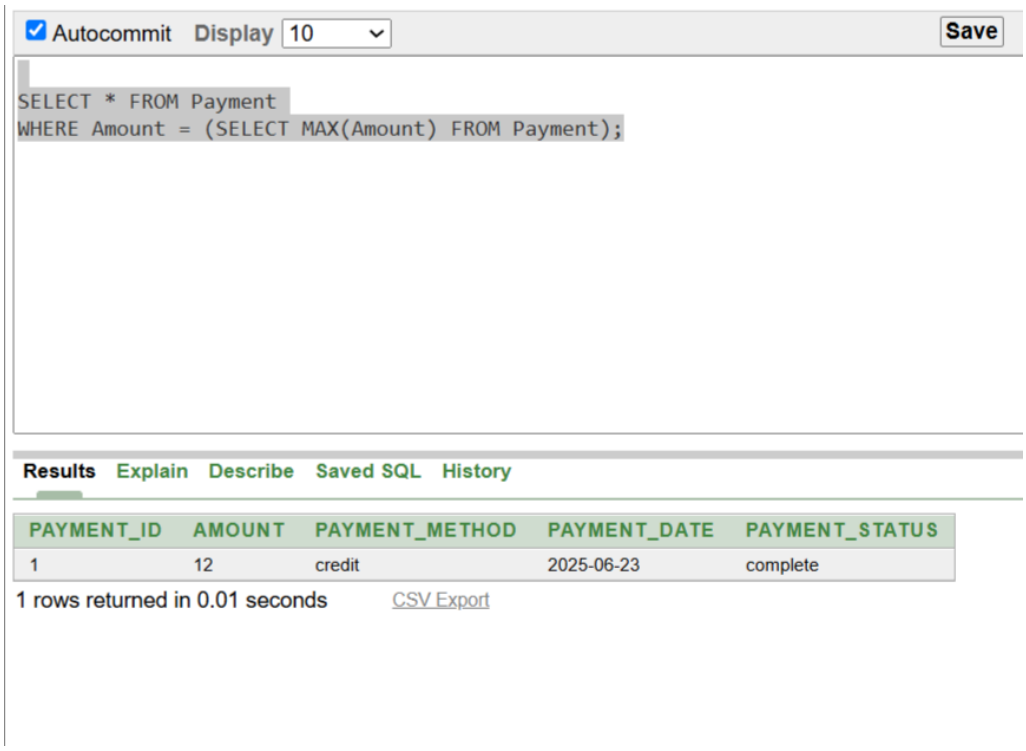
```
ayment ID 1 amount: 12

tatement processed.
```

Subquery:

**Payment with highest amount

SELECT * FROM Payment

WHERE Amount = (SELECT MAX(Amount) FROM Payment);



**

```
DECLARE
   v_payment_id Payment.Payment_ID%TYPE;
BEGIN
   FOR rec IN (
      SELECT Payment_ID
      FROM Payment
      WHERE Amount = (SELECT MAX(Amount) FROM Payment)
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Payment with highest amount: ' || rec.Payment_ID);
   END LOOP;
END;
/
```

Autocommit Display 10

```
DECLARE
    v_payment_id Payment.Payment_ID%TYPE;
BEGIN
    FOR rec IN (
        SELECT Payment_ID
        FROM Payment
        WHERE Amount = (SELECT MAX(Amount) FROM Payment)
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Payment with highest amount: ' || rec.Payment_ID);
    END LOOP;
END;
/
```

Results Explain Describe Saved SQL History

```
Payment with highest amount: 3

Statement processed.
```

0.00 seconds

Joining:

**Show baby info by gender

```
BEGIN
    FOR rec IN (
        SELECT B.Baby_ID, B.BabyName, B.BabyAge, B.BabyGender
        FROM Baby B
        ORDER BY B.BabyGender
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Baby ID: ' || rec.Baby_ID || ', Name: ' || rec.BabyName || ', Age: ' || rec.BabyAge || ', Gender: ' || rec.BabyGender);
    END LOOP;
END;
/
```

Autocommit Display 10 Save Run

```
-- Show baby info by gender
BEGIN
    FOR rec IN (
        SELECT B.Baby_ID, B.BabyName, B.BabyAge, B.BabyGender
        FROM Baby B
        ORDER BY B.BabyGender
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Baby ID: ' || rec.Baby_ID || ', Name: ' || rec.BabyName || ', Age: ' || rec.BabyAge || ', Gender: ' || rec.BabyGender);
    END LOOP;
END;
/
```

Results Explain Describe Saved SQL History

```
Baby ID: 1, Name: ju, Age: 12, Gender: MALE

Statement processed.

0.00 seconds
```

**

-- Show babysitter info by ID

```
BEGIN
   FOR rec IN (
      SELECT Babysitter_ID, BabySName, BabySPhone, Availability_Status, Hourly_Rate
      FROM Babysitter
      ORDER BY Babysitter_ID
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Babysitter ID: ' || rec.Babysitter_ID || ', Name: ' || rec.BabySName || ', Phone: ' || rec.BabySPhone || ', Status: ' || rec.Availability_Status || ', Rate: ' || rec.Hourly_Rate);
   END LOOP;
END;
/
```

Advance PL/SQL

Function:

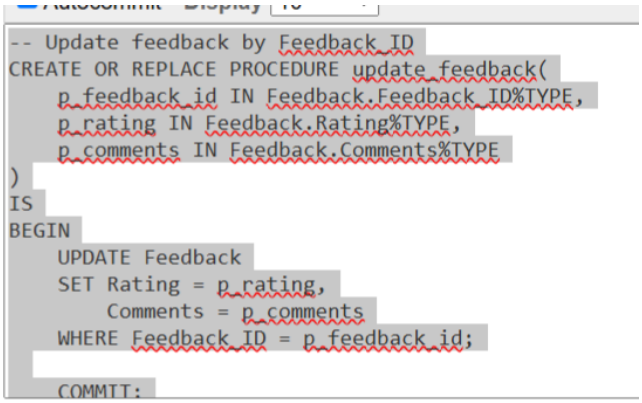
Total booking by Payment_ID

```
CREATE OR REPLACE FUNCTION total_booking_by_payment(p_payment_id IN
Booking.Payment_ID%TYPE)

RETURN NUMBER

IS

   v_total NUMBER := 0;

BEGIN

   SELECT NVL(COUNT(*), 0)

   INTO v_total

   FROM Booking

   WHERE Payment_ID = p_payment_id;


   RETURN v_total;

END;

/


BEGIN

   DBMS_OUTPUT.PUT_LINE('Total Booking: ' || total_booking_by_payment('1'));

END;

/
```

Procedure:

```
-- Update feedback by Feedback_ID
CREATE OR REPLACE PROCEDURE update_feedback(
    p_feedback_id IN Feedback.Feedback_ID%TYPE,
    p_rating IN Feedback.Rating%TYPE,
    p_comments IN Feedback.Comments%TYPE
)
IS
BEGIN
    UPDATE Feedback
    SET Rating = p_rating,
        Comments = p_comments
    WHERE Feedback_ID = p_feedback_id;

    COMMIT;
END;
/
```

```
BEGIN
    update_feedback(1, '5 Stars', 'good');
END;
```



/

**Identify babysitter by babysitter name

```
CREATE OR REPLACE PROCEDURE identify_babysitter_by_name(
    p_babysitter_id IN Babysitter.Babysitter_ID%TYPE,
    p_babysname IN Babysitter.BabySName%TYPE,
    p_babysphone IN Babysitter.BabySPhone%TYPE,
    p_availability_status IN Babysitter.Availability_Status%TYPE,
    p_hourly_rate IN Babysitter.Hourly_Rate%TYPE
)
IS
BEGIN
    INSERT INTO Babysitter (
        Babysitter_ID, BabySName, BabySPhone, Availability_Status, Hourly_Rate
    )
    VALUES (
        p_babysitter_id, p_babysname, p_babysphone, p_availability_status, p_hourly_rate
    );

    COMMIT;
END;
/
```

BEGIN

identify_babysitter_by_name('2', 'Sarah Ahmed', '01712345678', 'Available', 150.00);

END;

/

SELECT * FROM Babysitter;

```
CREATE OR REPLACE PROCEDURE identify_babysitter_by_name(
    p_babysitter_id IN Babysitter.Babysitter_ID%TYPE,
    p_babysname IN Babysitter.BabySName%TYPE,
    p_babysphone IN Babysitter.BabySPhone%TYPE,
    p_availability_status IN Babysitter.Availability_Status%TYPE,
    p_hourly_rate IN Babysitter.Hourly_Rate%TYPE
)
IS
BEGIN
    INSERT INTO Babysitter (
        Babysitter_ID, BabySName, BabySPhone, Availability_Status, Hourly_Rate
    )
    VALUES (
```

Results Explain Describe Saved SQL History

tatement processed.

.00 seconds

Table-Based Record:

**Show client details by CLIENT_ID

CREATE OR REPLACE PROCEDURE show_client_details(p_client_id IN CLIENT.CLIENT_ID%TYPE)

IS

v_client CLIENT%ROWTYPE;

BEGIN

SELECT * INTO v_client FROM CLIENT WHERE CLIENT_ID = p_client_id;

```
    DBMS_OUTPUT.PUT_LINE('Client ID: ' || v_client.CLIENT_ID);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_client.Name);
    DBMS_OUTPUT.PUT_LINE('Phone: ' || v_client.Phone);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_client.Email);
    DBMS_OUTPUT.PUT_LINE('Address: ' || v_client.Address);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Client not found.');
END;
/


BEGIN
    show_client_details('CL001');
END;
/
```

```
CREATE OR REPLACE PROCEDURE show_client_details(p_client_id IN CLIENT.CLIENT_ID%TYPE)
IS
    v_client CLIENT%ROWTYPE;
BEGIN
    SELECT * INTO v_client FROM CLIENT WHERE CLIENT_ID = p_client_id;

    DBMS_OUTPUT.PUT_LINE('Client ID: ' || v_client.CLIENT_ID);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_client.Name);
    DBMS_OUTPUT.PUT_LINE('Phone: ' || v_client.Phone);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_client.Email);
    DBMS_OUTPUT.PUT_LINE('Address: ' || v_client.Address);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
```

Results Explain Describe Saved SQL History

```
Client ID: 1
Name: ju
Phone: 12
Email: ju@
Address: ju

Statement processed.

0.00 seconds
```

Language: en-gb

**Show all babysitters by BabySName

```
CREATE OR REPLACE PROCEDURE list_babysitters_by_name(p_babysname IN Babysitter.BabySName%TYPE)
IS
   v_babysitter Babysitter%ROWTYPE;
BEGIN
   FOR rec IN (SELECT * FROM Babysitter WHERE BabySName = p_babysname) LOOP
      v_babysitter := rec;
      DBMS_OUTPUT.PUT_LINE('Babysitter ID: ' || v_babysitter.Babysitter_ID);
      DBMS_OUTPUT.PUT_LINE('Name: ' || v_babysitter.BabySName);
      DBMS_OUTPUT.PUT_LINE('Phone: ' || v_babysitter.BabySPhone);
      DBMS_OUTPUT.PUT_LINE('Availability: ' || v_babysitter.Availability_Status);
      DBMS_OUTPUT.PUT_LINE('Hourly Rate: ' || v_babysitter.Hourly_Rate);
      DBMS_OUTPUT.PUT_LINE('-----------------------------');
   END LOOP;
END;
/

BEGIN
   list_babysitters_by_name('Sarah Ahmed');
END;
```

/

```
CREATE OR REPLACE PROCEDURE list_babysitters_by_name(p_babysname IN Babysitter.BabySName%TYPE)
IS
    v_babysitter Babysitter%ROWTYPE;
BEGIN
    FOR rec IN (SELECT * FROM Babysitter WHERE BabySName = p_babysname) LOOP
        v_babysitter := rec;
        DBMS_OUTPUT.PUT_LINE('Babysitter ID: ' || v_babysitter.Babysitter_ID);
        DBMS_OUTPUT.PUT_LINE('Name: ' || v_babysitter.BabySName);
        DBMS_OUTPUT.PUT_LINE('Phone: ' || v_babysitter.BabySPhone);
        DBMS_OUTPUT.PUT_LINE('Availability: ' || v_babysitter.Availability_Status);
        DBMS_OUTPUT.PUT_LINE('Hourly Rate: ' || v_babysitter.Hourly_Rate);
        DBMS_OUTPUT.PUT_LINE('----------------------------');
    END LOOP;
```

Results Explain Describe Saved SQL History

```
Babysitter ID: 2
Name: Sarah Ahmed
Phone: 01712345678
Availability: Available
Hourly Rate: 150
----------------------------

Statement processed.
```

0.00 seconds