

**MYMZ TEAM**

**Bus Station Database Project**

***Instructor :Dr.raghda hriez***

***Students :Yousef jarbou 20200306***

***Mohammad Inshasi 20200841***

***Zaid abdullah 20200284***

***Mohammad kharroub 20200244***

* **Description:**

The Bus Station Database Project aims to create a comprehensive and efficient system for managing the operations of a bus station. The project will develop a database that encompasses various entities such as persons, users, drivers, tickets, buses, routes, and checkpoints. The database will facilitate ticket booking, user management, bus allocation, and route tracking functionalities.

The core entities in the database include "Person," which represents individuals associated with the bus station. A person is identified by their ID and has attributes such as name (first name and last name), birthdate, age, email, password, and mobile number. Persons can either be classified as "Users" or "Drivers," but not both simultaneously. Users have an additional attribute of an ID, while drivers have their driving license type recorded.

Tickets play a crucial role in the system and are categorized as "Economy" or "VIP." Each ticket is assigned a unique ID and features attributes such as price, status, departure time, expiry date, and activation date. Economy tickets include a discount, while VIP tickets provide meals during the journey.

Buses are represented by their ID and a corresponding number. Each bus is assigned a driver, referenced by the driver's ID, and travels through a specific route. The route entity consists of an ID, pick-up point, arrival point, and may have multiple checkpoints along the way.

The database also considers relationships between entities. Users can act as parents to other users, creating a hierarchical structure. Each user can book one or more tickets, and a ticket can only be booked by one user. Buses are associated with tickets, where each ticket belongs to a single bus. Drivers are assigned to drive multiple buses, and each bus is driven by one driver. Furthermore, buses travel through specific routes, and each route can be traveled through by multiple buses.

By implementing the Bus Station Database Project, the bus station will benefit from streamlined ticket booking processes, efficient user management, accurate bus allocation, and effective route tracking. The database system will enhance overall operational efficiency, resulting in improved service delivery and customer satisfaction.

* **Requirements as (Tables):**

1. \*\*Person Table\*\*:

- ID (Primary Key)

- First Name

- Last Name

- Birthdate

- Age

- Email

- Password

- Mobile Number

2. \*\*User Table\*\*:

- ID (Foreign Key referencing Person)

- Parent ID (Foreign Key referencing User)

3. \*\*Driver Table\*\*:

- ID (Foreign Key referencing Person)

- Driving License Type

4. \*\*Ticket Table\*\*:

- ID (Primary Key)

- Price

- Status

- Departure Time

- Expiry Date

- Activation Date

- Type (Economy or VIP)

- Discount (only for economy tickets)

- Meals (only for VIP tickets)

5. \*\*Bus Table\*\*:

- ID

- Number

- Driver ID (Foreign Key referencing Driver)

- Route ID (Foreign Key referencing Route)

6. \*\*Route Table\*\*:

- ID

- Pick-up Point

- Arrival Point

7. \*\*Checkpoint Table\*\*:

- ID

- Route ID (Foreign Key referencing Route)

- Checkpoint Name

* **Relationships:**

- User (Parent) to User (Child): Each user can have one parent and multiple children.

- User to Ticket: Each user can book one or more tickets.

- Ticket to User: Each ticket can be booked by one user.

- Ticket to Bus: Each ticket belongs to one bus.

- Bus to Ticket: Each bus can have multiple tickets.

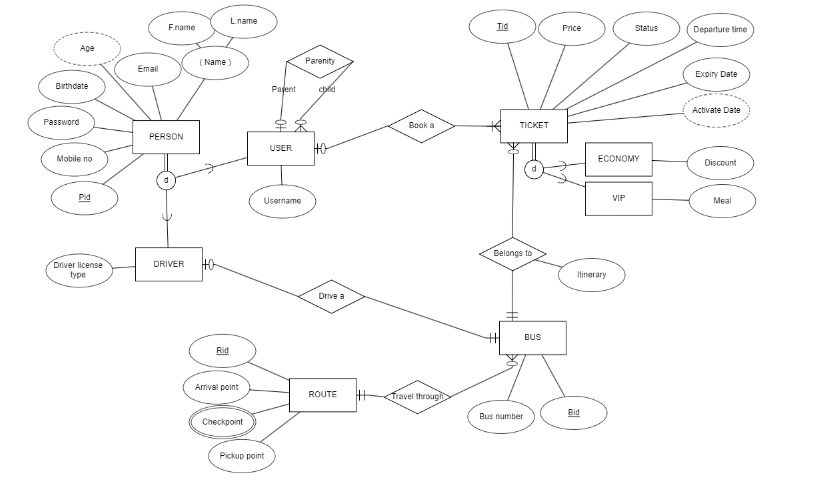
- Driver to Bus: Each driver can drive multiple buses.

- Bus to Driver: Each bus is driven by one driver.

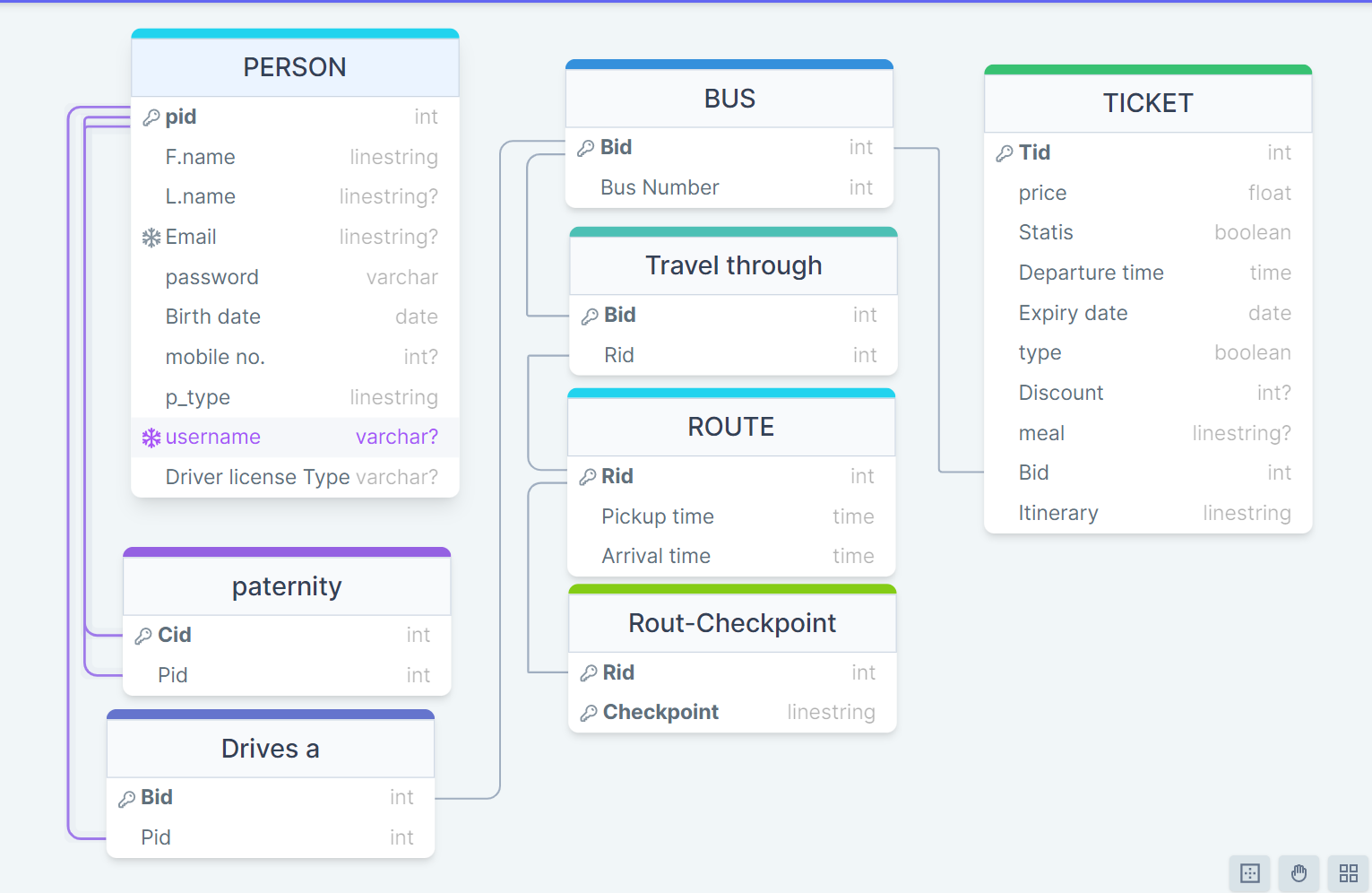
- Bus to Route: Each bus travels through one route.

- Route to Bus: Each route can be traveled through by multiple buses.

* **ER-diagram:**

****

* **Schema:**

****

* **SQL code :**

CREATE TABLE ROUTE (

Rid INT NOT NULL,

Pickup\_point VARCHAR2(100) NOT NULL,

Arrival\_point VARCHAR2(100) NOT NULL,

PRIMARY KEY (Rid)

);

CREATE TABLE ROUTE\_CHECKPOINT (

Checkpoint VARCHAR2(100) NOT NULL,

Rid INT NOT NULL,

PRIMARY KEY (Checkpoint, Rid),

FOREIGN KEY (Rid) REFERENCES ROUTE(Rid)

ON DELETE CASCADE

);

CREATE TABLE USERS (

PID NUMBER,

FNAME VARCHAR2(50),

LNAME VARCHAR2(50),

EMAIL VARCHAR2(100),

BIRTHDATE DATE,

PASSWORD VARCHAR2(100),

MOBILE\_NO VARCHAR2(20),

PARENT\_PID NUMBER,

CONSTRAINT PK\_USERS PRIMARY KEY (PID),

CONSTRAINT FK\_USERS\_PARENT FOREIGN KEY (PARENT\_PID) REFERENCES USERS (PID)

ON DELETE CASCADE

);

CREATE TABLE BUS (

Bid INT NOT NULL,

Rid INT NOT NULL,

PRIMARY KEY (Bid),

FOREIGN KEY (Rid) REFERENCES ROUTE(Rid)

ON DELETE CASCADE

);

CREATE TABLE TICKET (

Tid INT NOT NULL,

Expiry\_date DATE NOT NULL,

Status VARCHAR2(20) NOT NULL,

Departure\_time DATE NOT NULL,

Pid INT,

Bid INT NOT NULL,

PRIMARY KEY (Tid),

FOREIGN KEY (Pid) REFERENCES USERS(PID),

FOREIGN KEY (Bid) REFERENCES BUS(Bid)

ON DELETE CASCADE

);

CREATE TABLE DRIVER

(

Pid INT NOT NULL,

License\_no VARCHAR2(100) NOT NULL,

FNAME VARCHAR2(50),

LNAME VARCHAR2(50),

Bid INT,

PRIMARY KEY (Pid),

FOREIGN KEY (Pid) REFERENCES USERS(PID),

FOREIGN KEY (Bid) REFERENCES BUS(Bid)

ON DELETE CASCADE

);

CREATE TABLE VIP (

Tid INT NOT NULL,

V\_Price FLOAT NOT NULL,

PRIMARY KEY (Tid),

FOREIGN KEY (Tid) REFERENCES TICKET(Tid)

ON DELETE CASCADE

);

CREATE TABLE ECONOMY (

Tid INT NOT NULL,

E\_Price FLOAT NOT NULL,

PRIMARY KEY (Tid),

FOREIGN KEY (Tid) REFERENCES TICKET(Tid)

ON DELETE CASCADE

);

* **SQL CODE (INSERT)**

-- Inserting data into the ROUTE table

INSERT INTO ROUTE (RID, PICKUP\_POINT, ARRIVAL\_POINT)

VALUES (1, 'New York', 'Washington DC');

-- Inserting data into the USERS table

INSERT INTO USERS (PID, FNAME, LNAME, EMAIL, BIRTHDATE, PASSWORD, MOBILE\_NO)

VALUES (1, 'Zaid', 'Abdullah', 'ZaidAbdullah@psut.com', TO\_DATE('2002-09-01', 'YYYY-MM-DD'), 'password123', '1234567890');

-- Inserting data into the BUS table

INSERT INTO BUS (Bid, Rid)

VALUES (1, 1);

-- Inserting data into the TICKET table

INSERT INTO TICKET (Tid, Expiry\_date, Status, Departure\_time, Pid, Bid)

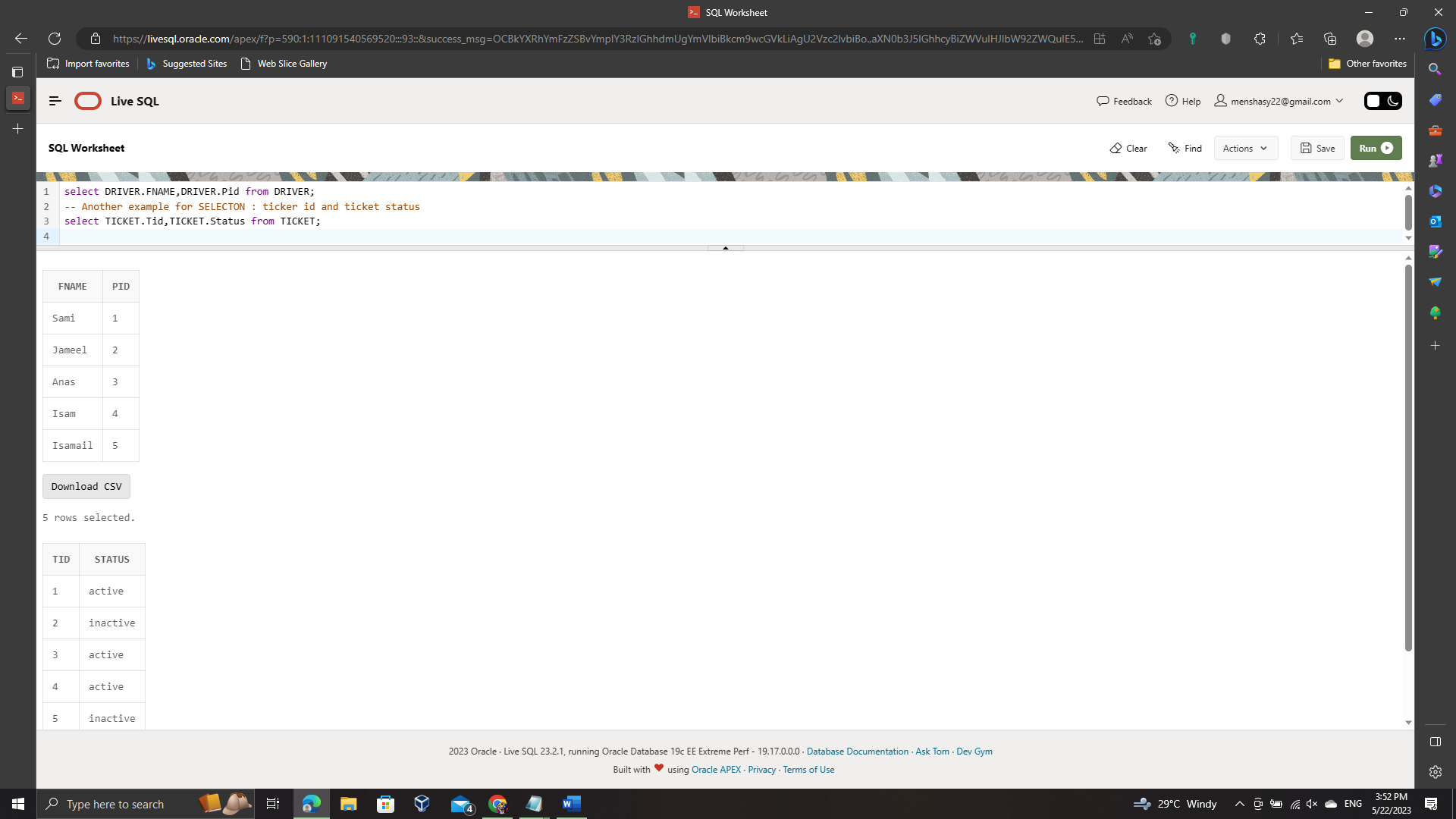
VALUES (1, TO\_DATE('2023-05-10', 'YYYY-MM-DD'), 'active', TO\_DATE('2023-05-15 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1, 1);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* **SQL CODE (SELECT)+SS FROM SQL LIVE**

–select DRIVER.FNAME,DRIVER.Pid from DRIVER;

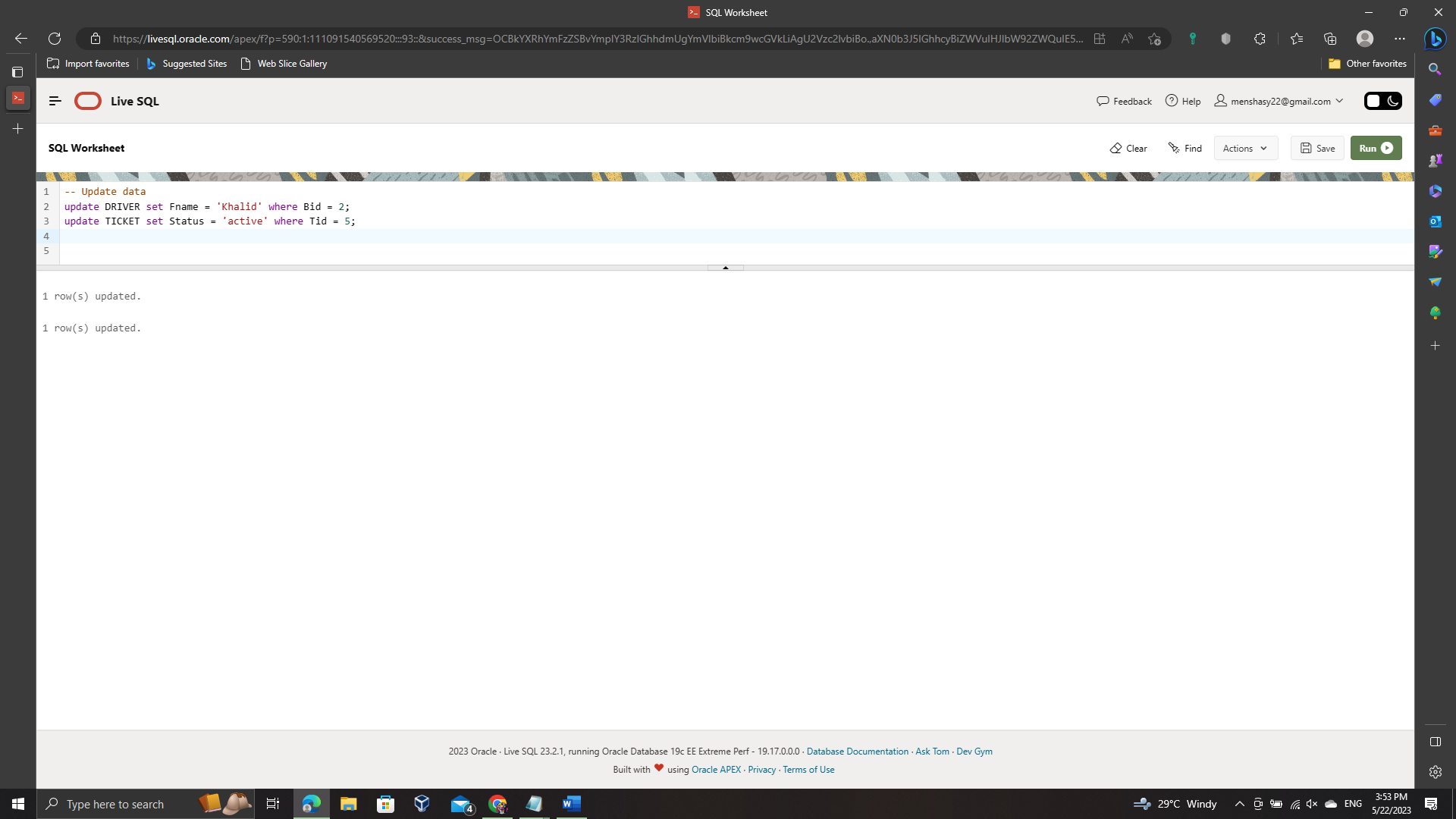
–select TICKET.Tid,TICKET.Status from TICKET;

****

* **SQL CODE (UPDATE)+SS FROM SQL LIVE**

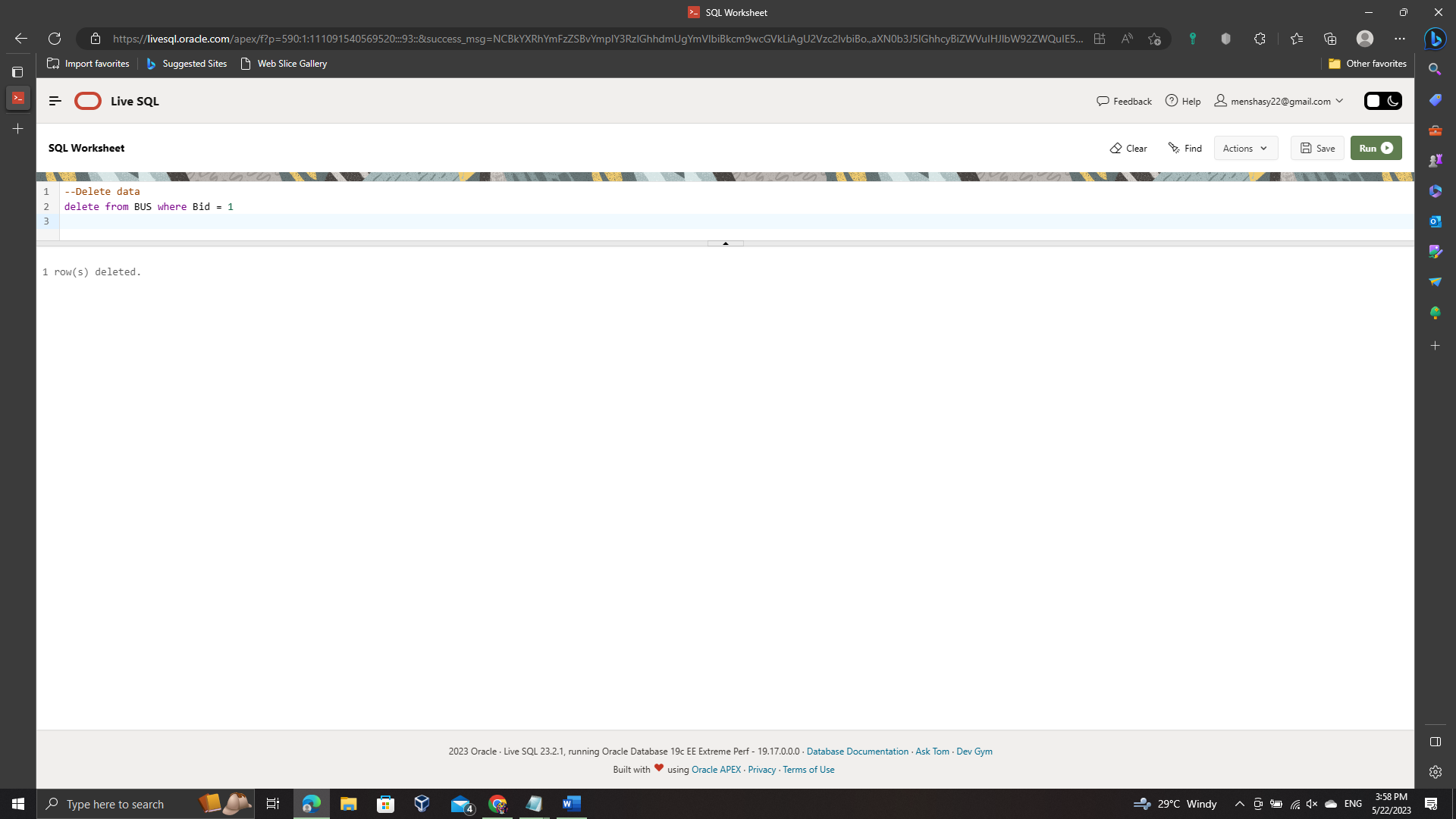
–update Driver set Fname = 'Khalid' where Bid = 2;

–update TICKET set Status = 'active' where Tid = 5;

****

* **SQL CODE (DELETE)+SS FROM SQL LIVE**

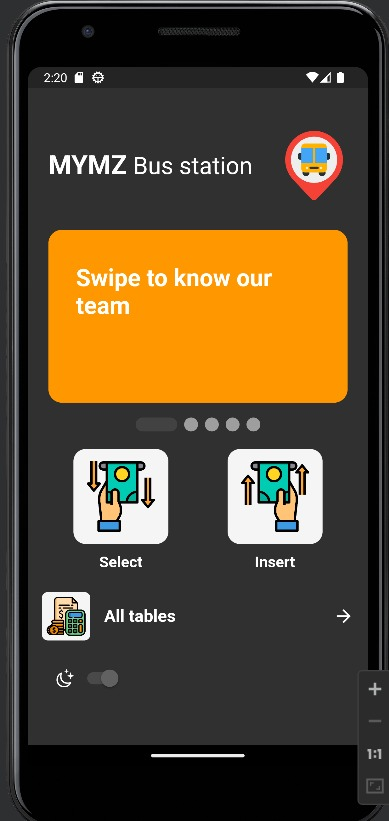
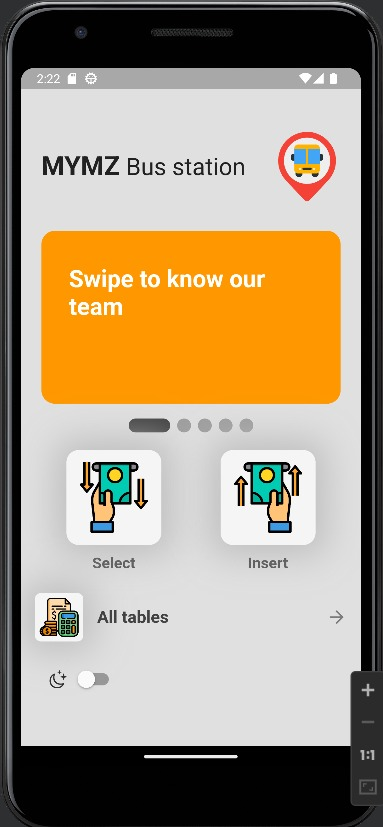
–delete from BUS where Bid = 1;

****

* **Flutter Interface**

In this era where mobile phone applications dominate users' interests, our project aims to develop a mobile application that caters to a wide range of devices. To achieve this, we have chosen to utilize a multiplatform framework called Flutter, developed by Google and powered by the Dart programming language.

One of our primary objectives in designing this application is to create a user-friendly and easily understandable interface. We have incorporated various elements to enhance the user experience. For instance, we have implemented the "Page View" widget, which allows users to swipe through and familiarize themselves with our team members. Additionally, we have included buttons for inserting, selecting, and viewing all tables within the application. To cater to user preferences, we have even added a delightful feature called "Daaark Moood," which enables users to switch to a dark theme for enhanced visual comfort.



To provide a seamless experience, we understand the importance of preserving user data even when the application is closed and reopened. Consequently, we have incorporated a database within the application. There are multiple solutions available for implementing databases, such as NoSQL or SQL-based solutions. In our case, we have chosen to leverage an SQL-based approach.

Integrating SQL code into Flutter required us to add a package called SQLite to the project's files. Once this package was integrated, we were able to create a dedicated class that encompasses SQL table creations, insert queries, select, update, and delete operations. To simplify data retrieval, we have also implemented convenient getters that facilitate interaction with the stored data.

We believe that our chosen framework, Flutter, combined with the utilization of an SQL database, will enable us to deliver a robust and user-friendly mobile application that can run smoothly across various devices. To get a comprehensive understanding of the features and functionality of our application, we highly recommend watching the accompanying video, which showcases the app's amazing capabilities.

