TAXI SERVICE DATABASE MANAGEMENT SYSTEM

UCS 310 Database Management System Project Report

END-Semester Evaluation

Submitted by:

(102203388) AKSHAT GUPTA

(102203440) ASMI JUNEJA

(102203446) ISHANT VINAIK

(102203533) PREETISH MANGI

BE Second Year, COE

Submitted to:

ARCHANA SINGH

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

TIET, Patiala

May 2024

INDEX

S.NO	TOPIC	PAGE NO,
1	PROJECT OBJECTIVE	3
2	RELATIONAL SCHEMA	6
3	ER DIAGRAM	7
4	ER TO TABLE	8
5	NORMALISATION	10
6	SQL CODES WITH SCREENSHOTS	11
	• CREATE AND INSERT FUNCTIONS	11
	• CONSTRAINTS	23
	• TRIGGERS	24
	• PROCEDURES	27
	• FUNCTIONS	31
	• CURSOR	35
	• EXCEPTIONS	39
7	QUERIES	42
8	CONCLUSION	47
9	REFERENCES	48

PROJECT OBJECTIVE:

The Taxi Service Database Management System is designed to streamline the operations of an inter-city travel service provided by a taxi company. Users can register with the system, providing essential details such as name, contact information, and email address, each receiving a unique User ID for identification. Meanwhile, drivers are registered within the system with their particulars, including name, contact information, gender, and age, each assigned a unique Drive ID for identification. The system maintains comprehensive information about each taxi, including its unique Taxi ID, registration number, model, etc. Users can seamlessly book taxis for specified durations, recording trip start times and updating trip end times upon conclusion, generating unique Trip ID identifiers for each trip. Billing is efficiently managed, with unique bills generated for each trip containing user and trip details. Users can also provide feedback and ratings for trips, aiding in quality control and improvement, managed by customer service representatives. Overall, the system, overseen by an administrator, ensures smooth operations and enhances the overall user experience of the taxi service company.

REQUIREMENT ANALYSIS

Focusing on the needs and challenges faced in real-life scenarios encountered by a Taxi Database Management System, here is a detailed requirement analysis for the same:

1. **User Registration and Authentication:**
    - Requirement: Users should be able to register for the service by providing their personal information such as name, contact details, and payment information. Upon registration, users should be authenticated to access the system.
    - Need: Registration and authentication are essential for ensuring the security and integrity of the system. It allows the system to track and manage user accounts, as well as provide personalized services and maintain accountability for transactions.

2. **Real-Time Taxi Booking:**
    - Requirement: Users should be able to book taxis in real-time through the application. They should be able to specify their current location, destination, preferred vehicle type, and other preferences.
    - Need: Real-time taxi booking provides convenience and flexibility to users, allowing them to quickly and easily find transportation options tailored to their needs. It reduces wait times and uncertainty, improving overall user experience.

3. **Driver Management:**
    - Requirement: The system should manage a database of registered drivers, including their personal information, vehicle details, licensing and insurance information, and availability status.
    - Need: Effective driver management ensures that there is a sufficient pool of drivers available to meet the demand for taxi services. It allows the system to match drivers

with passengers based on proximity, availability, and other factors, optimizing the efficiency of the service.

4. **Route Optimization and Navigation:**
   - Requirement: The system should include route optimization and navigation features to help drivers efficiently navigate to their destinations. It should consider factors such as traffic conditions, road closures, and user preferences.
   - Need: Route optimization and navigation improve driver efficiency and reduce travel time, leading to faster pickups and drop-offs for passengers. It also helps minimize fuel consumption and vehicle wear and tear, leading to cost savings for drivers and operators.

5. **Payment Processing:**
   - Requirement: The system should support secure payment processing, allowing users to pay for their rides electronically through credit/debit cards, digital wallets, or other payment methods.
   - Need: Secure and convenient payment processing is essential for facilitating transactions between passengers and drivers. It ensures that payments are processed accurately and transparently, reducing the risk of disputes and fraud.

6. **Feedback and Rating System:**
   - Requirement: The system should include a feedback and rating system that allows users to rate their ride experience and provide feedback on drivers and the service.
   - Need: Feedback and rating systems help maintain service quality and accountability. They provide valuable insights into customer satisfaction and driver performance, allowing operators to identify areas for improvement and address any issues promptly.

7. **Administrative Tools and Reporting:**
   - Requirement: The system should include administrative tools and reporting features for operators to monitor and manage the service, including tracking bookings, managing driver accounts, generating reports, and analyzing performance metrics.
   - Need: Administrative tools and reporting capabilities are essential for operators to effectively manage and optimize the service. They provide visibility into key performance indicators, allowing operators to make informed decisions and take proactive measures to enhance the overall efficiency and profitability of the service.

In summary, a taxi database management system is needed in real life to address the complex requirements and challenges associated with operating a taxi service. By effectively managing user registrations, bookings, driver resources, route optimization, payment processing, feedback mechanisms, and administrative tasks, such a system can provide a seamless and reliable transportation experience for users while maximizing operational efficiency and profitability for operators.

Data Query Language (DQL)
SELECT-Used to retrieve certain records from one or more tables.

Data Manipulation Language (DML)
INSERT - Used to create a record
UPDATE - Used to change certain records
DELETE - Used to delete certain records.

Data Definition Language (DDL)
CREATE - Used to create a new table, a view of a table, or other object in database
ALTER - Used to modify an existing database object, such as a table
DROP - Used to delete an entire table, a view of a table or other object in the database

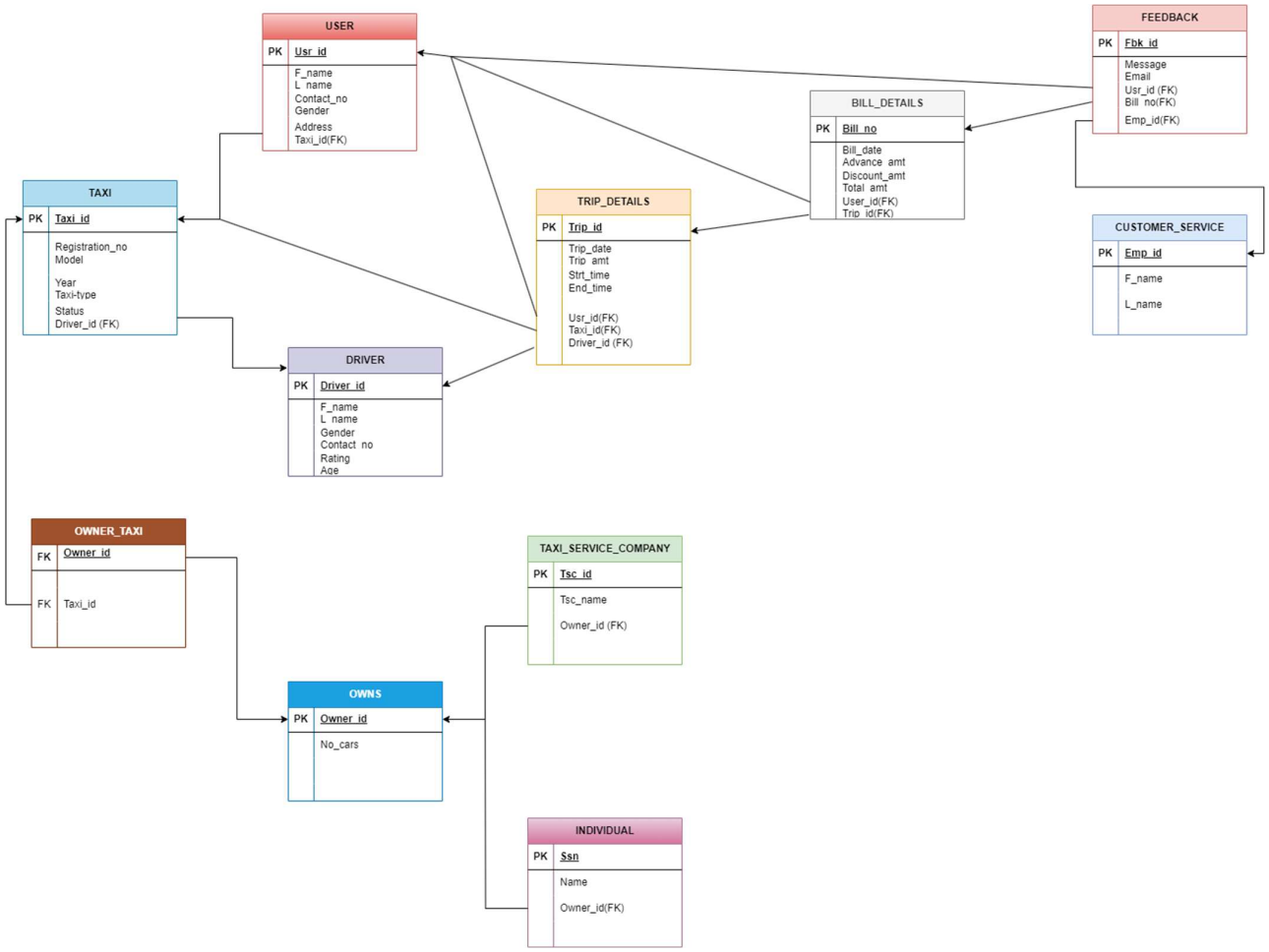
SOFTWARE REQUIREMENTS

Operating System: 64bit WINDOWS Operating System, X64- based processor
Database: MYSQL

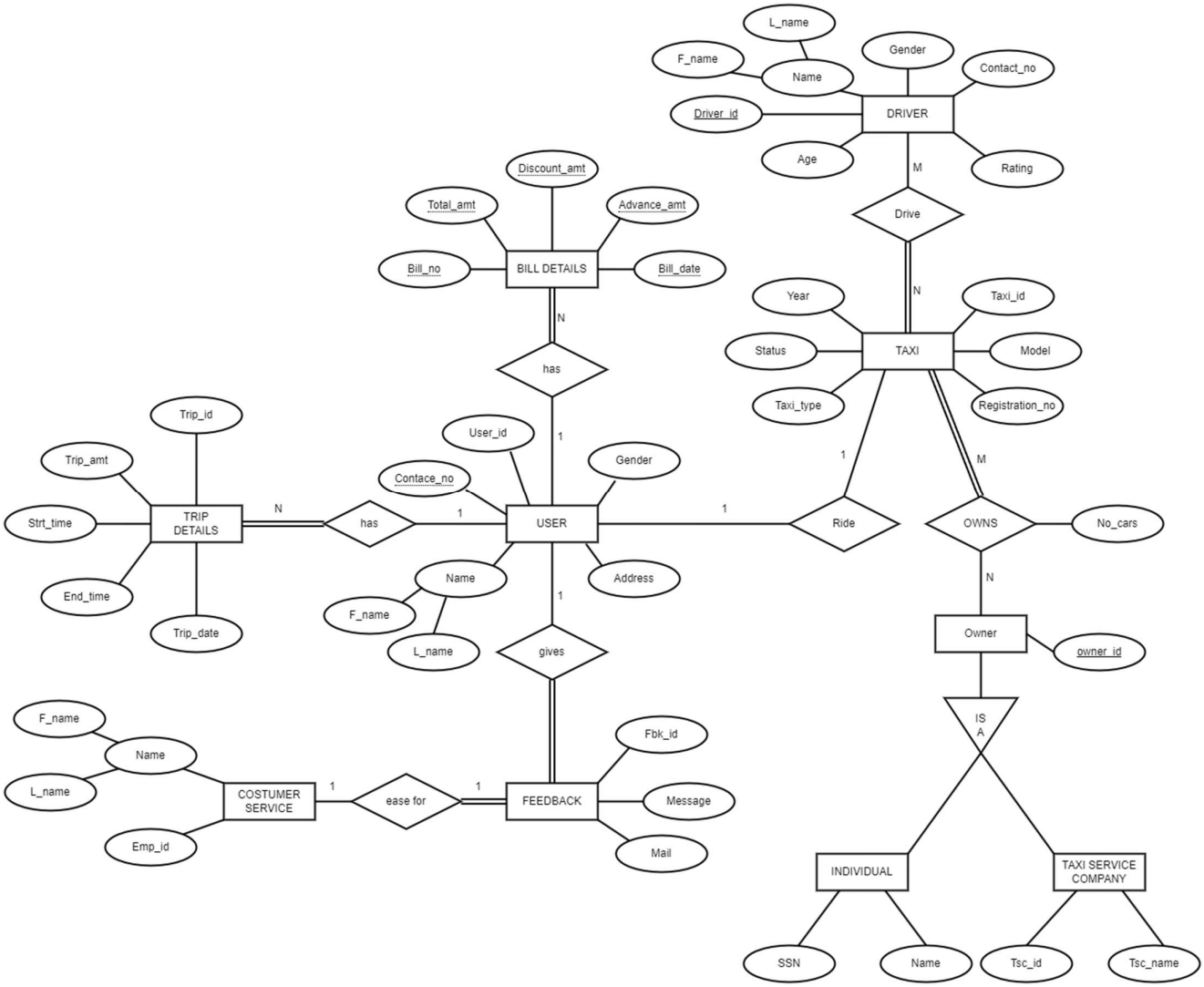
HARDWARE REQUIREMENTS

Processor: Intel Celeron CPU N3060 @1.60GHz or Above
RAM: 4.00 GB or Above
Hard Disk: 1 TB
Compact Disk: CD-ROM, CD-R, CD-RW
Input device: Keyboard

RELATIONAL SCHEMA:



ER DIAGRAM:


L_name
F_name
Name
Gender
Contact_no
Driver_id
DRIVER
Age
Rating
M
Drive
N
Discount_amt
Total_amt
Advance_amt
Bill_no
BILL DETAILS
Bill_date
N
has
1
Year
Taxi_id
Status
TAXI
Model
Taxi_type
Registration_no
Trip_id
User_id
Gender
Trip_amt
Contace_no
Strt_time
TRIP DETAILS
N
has
1
USER
1
Ride
1
M
OWNS
No_cars
End_time
Name
Address
F_name
1
Trip_date
L_name
gives
N
Owner
owner_id
IS A
F_name
Name
Fbk_id
L_name
COSTUMER SERVICE
1
ease for
1
FEEDBACK
Message
Emp_id
Mail
INDIVIDUAL
TAXI SERVICE COMPANY
SSN
Name
Tsc_id
Tsc_name

ER TO TABLE:

TAXI

Taxi_id	Registration_no	Taxi_Model	Taxi_Year	Taxi_type	Status	Driver_id

- Primary Key: Taxi_id
- Foreign Keys: Driver_id

USER_TBL

Usr_id	F_name	L_name	Contat_no	Gender	Address	Taxi_id

- Primary Key: Usr_id
- Foreign Keys: Taxi_id

DRIVER

Driver_id	F_name	L_name	Gender	Conatct_no	Rating	Age

- Primary Key: Driver_id
- Foreign Keys: NA

TRIP_DETAILS

Trip_id	Trip_date	Trip_amt	Driver_id	Usr_id

Taxi_id	Strt_time	End_time

- Primary Key: Trip_id
- Foreign Keys: Taxi_id, Usr_id, Driver_id

BILL_DETAILS

Bill_no	Bill_date	Advance_amt	Discount_amt	Total_amt	Usr_id	Trip_id

- Primary Key: Bill_no
- Foreign Keys: Usr_id, Trip_id

CUSTOMER_SERVICE

Emp_id	F_name	L_name

- Primary Key: Emp_id
- Foreign Keys: NA

FEEDBACK

Fbk_id	Message	Email	Emp_id	Usr_id	Trip_id

- Primary Key: Fbk_id
- Foreign Keys: Usr_id, Emp_id, Trip_id

OWNER_TAXI

Owner_id	Taxi_id

- Primary Key: Owner_id, Taxi_id
- Foreign Keys: Owner_id, Taxi_id

OWNS

Owner_id	No_Cars

- Primary Key: Owner_id
- Foreign Keys: NA

INDIVIDUAL

Ssn	Name	Owner_id

- Primary Key: Ssn
- Foreign Keys: Owner_id

TAXI_SERVICE_COMPANY

Tsc_id	Tsc_name	Owner_id

- Primary Key: Tsc_id
- Foreign Keys: Owner_id

NORMALIZATION

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

First Normal Form (1NF)

- All tables have a primary key
- No repeating groups or arrays
- Each column value is atomic

All tables in the given schema are already in 1NF.

Second Normal Form (2 NF)

- Already in 1NF
- No partial dependency (no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table)

All tables in the given schema are already in 2NF.

Third Normal Form (3 NF)

- Already in 2NF
- No transitive dependency (If x dependent on Prime attribute and further non-prime attribute dependent on x, such that non-prime attribute is dependent on Prime attribute)

All tables in the given schema are already in 3NF.

Boyce – Coded Normal Form (BCNF)

- Already in 3NF
- Every functional dependency $X \rightarrow Y$ in a table, X should be a super key of that table

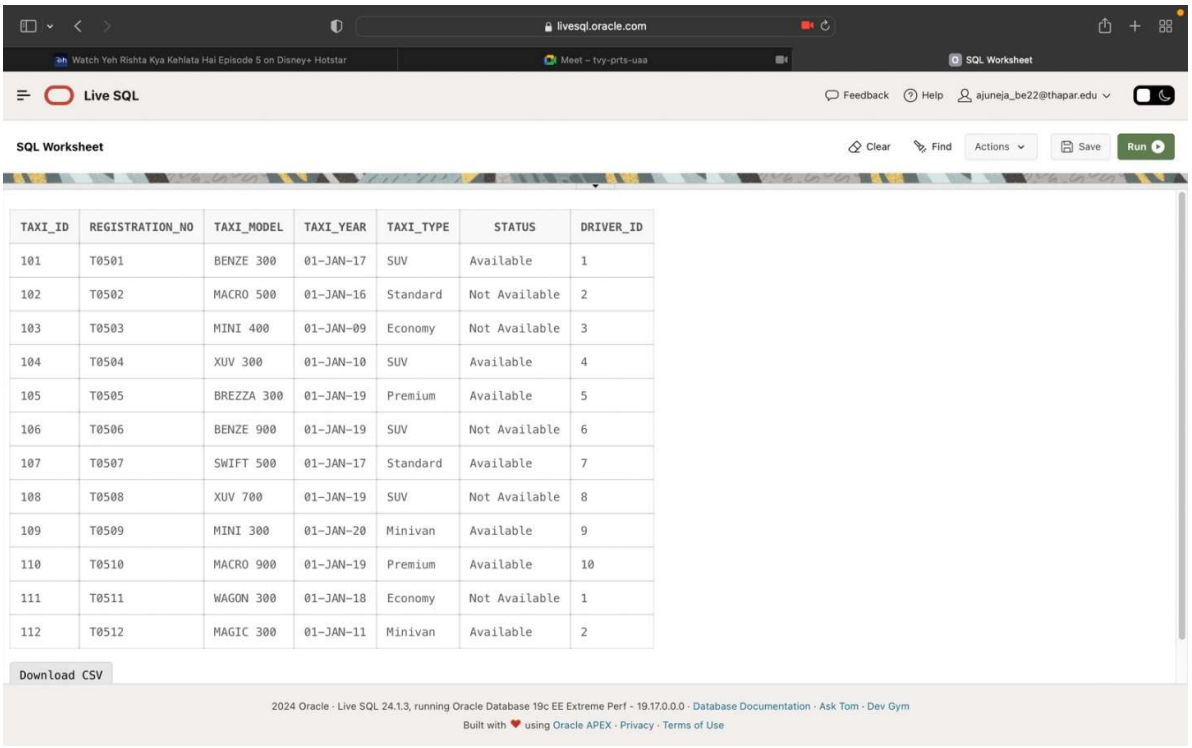
All tables in the given schema are already in BCNF.

SCREENSHOTS WITH OUTPUTS

A) CREATION AND INSERTION OF TABLES

CREATION AND INSERTION OF TAXI

```
CREATE TABLE TAXI (
  Taxi_id integer NOT NULL,
  Registration_no VARCHAR(20),
  Taxi_Model VARCHAR(20),
  Taxi_Year DATE,
  Taxi_type VARCHAR(20),
  Status VARCHAR(20),
  Driver_id integer,
  PRIMARY KEY (Taxi_id),
  UNIQUE (Registration_no)
);

INSERT INTO TAXI VALUES(101,'T0501','BENZE
300',to_date('01/01/2017','dd/mm/yyyy'),'SUV','Available',1);
INSERT INTO TAXI VALUES(102,'T0502','MACRO
500',to_date('01/01/2016','dd/mm/yyyy'),'Standard','Not Available',2);
INSERT INTO TAXI VALUES(103,'T0503','MINI
400',to_date('01/01/2009','dd/mm/yyyy'),'Economy','Not Available',3);
INSERT INTO TAXI VALUES(104,'T0504','XUV
300',to_date('01/01/2010','dd/mm/yyyy'),'SUV','Available',4);
INSERT INTO TAXI VALUES(105,'T0505','BREZZA
300',to_date('01/01/2019','dd/mm/yyyy'),'Premium','Available',5);
INSERT INTO TAXI VALUES(106,'T0506','BENZE
900',to_date('01/01/2019','dd/mm/yyyy'),'SUV','Not Available',6);
INSERT INTO TAXI VALUES(107,'T0507','SWIFT
500',to_date('01/01/2017','dd/mm/yyyy'),'Standard','Available',7);
INSERT INTO TAXI VALUES(108,'T0508','XUV
700',to_date('01/01/2019','dd/mm/yyyy'),'SUV','Not Available',8);
INSERT INTO TAXI VALUES(109,'T0509','MINI
300',to_date('01/01/2020','dd/mm/yyyy'),'Minivan','Available',9);
INSERT INTO TAXI VALUES(110,'T0510','MACRO
900',to_date('01/01/2019','dd/mm/yyyy'),'Premium','Available',10);
INSERT INTO TAXI VALUES(111,'T0511','WAGON
300',to_date('01/01/2018','dd/mm/yyyy'),'Economy','Not Available',1);
INSERT INTO TAXI VALUES(112,'T0512','MAGIC
300',to_date('01/01/2011','dd/mm/yyyy'),'Minivan','Available',2);
```

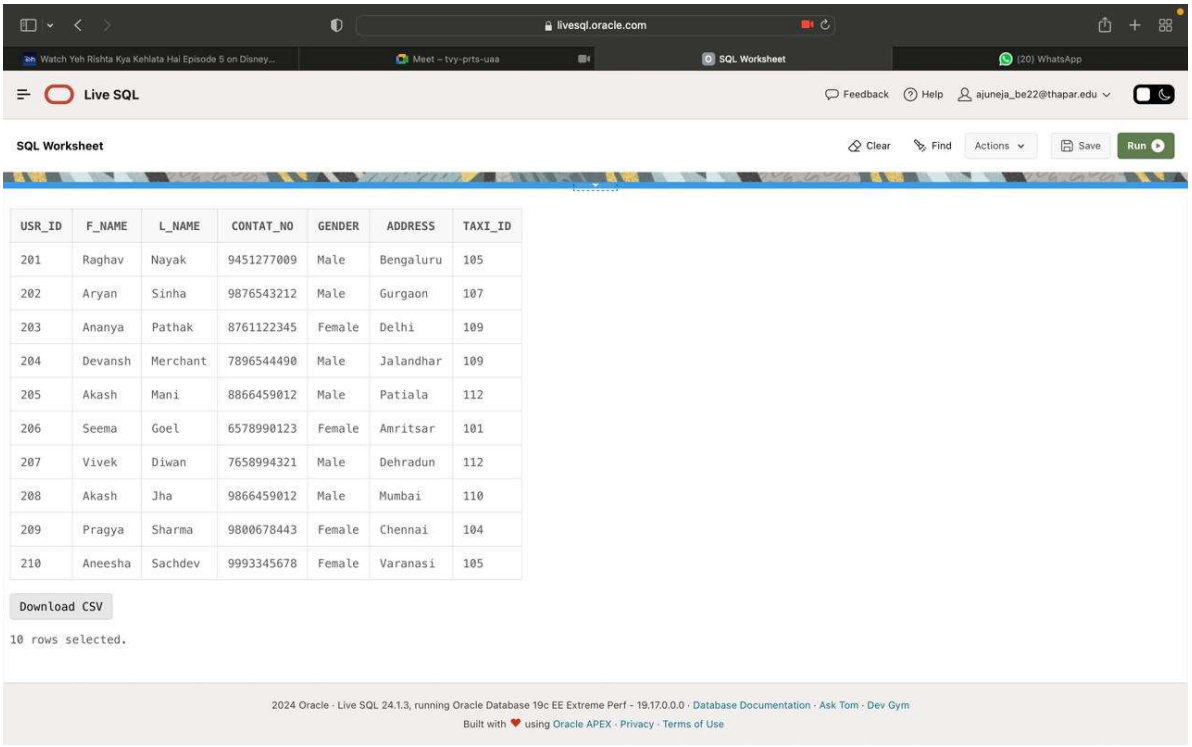


TAXI_ID	REGISTRATION_NO	TAXI_MODEL	TAXI_YEAR	TAXI_TYPE	STATUS	DRIVER_ID
101	T0501	BENZE 300	01-JAN-17	SUV	Available	1
102	T0502	MACRO 500	01-JAN-16	Standard	Not Available	2
103	T0503	MINI 400	01-JAN-09	Economy	Not Available	3
104	T0504	XUV 300	01-JAN-10	SUV	Available	4
105	T0505	BREZZA 300	01-JAN-19	Premium	Available	5
106	T0506	BENZE 900	01-JAN-19	SUV	Not Available	6
107	T0507	SWIFT 500	01-JAN-17	Standard	Available	7
108	T0508	XUV 700	01-JAN-19	SUV	Not Available	8
109	T0509	MINI 300	01-JAN-20	Minivan	Available	9
110	T0510	MACRO 900	01-JAN-19	Premium	Available	10
111	T0511	WAGON 300	01-JAN-18	Economy	Not Available	1
112	T0512	MAGIC 300	01-JAN-11	Minivan	Available	2

CREATION AND INSERTION OF USER

```
CREATE TABLE  USER_TBL (
  Usr_id integer NOT NULL,
  F_name VARCHAR(20),
  L_name VARCHAR(20),
  Contat_no integer,
  Gender VARCHAR(10),
  Address VARCHAR(50),
  Taxi_id integer,
  PRIMARY KEY (Usr_id)
);

INSERT INTO USER_TBL
VALUES(201,'Raghav','Nayak','9451277009','Male','Bengaluru','105');
INSERT INTO USER_TBL
VALUES(202,'Aryan','Sinha','9876543212','Male','Gurgaon','107');
INSERT INTO USER_TBL
VALUES(203,'Ananya','Pathak','8761122345','Female','Delhi','109');
INSERT INTO USER_TBL
VALUES(204,'Devansh','Merchant','7896544490','Male','Jalandhar','109');
INSERT INTO USER_TBL
VALUES(205,'Akash','Mani','8866459012','Male','Patiala','112');
```

```
INSERT INTO USER_TBL
VALUES(206,'Seema','Goel','6578990123','Female','Amritsar','101');
INSERT INTO USER_TBL
VALUES(207,'Vivek','Diwan','7658994321','Male','Dehradun','112');
INSERT INTO USER_TBL
VALUES(208,'Akash','Jha','9866459012','Male','Mumbai','110');
INSERT INTO USER_TBL
VALUES(209,'Pragya','Sharma','9800678443','Female','Chennai','104');
INSERT INTO USER_TBL
VALUES(210,'Aneesha','Sachdev','9993345678','Female','Varanasi','105');
```

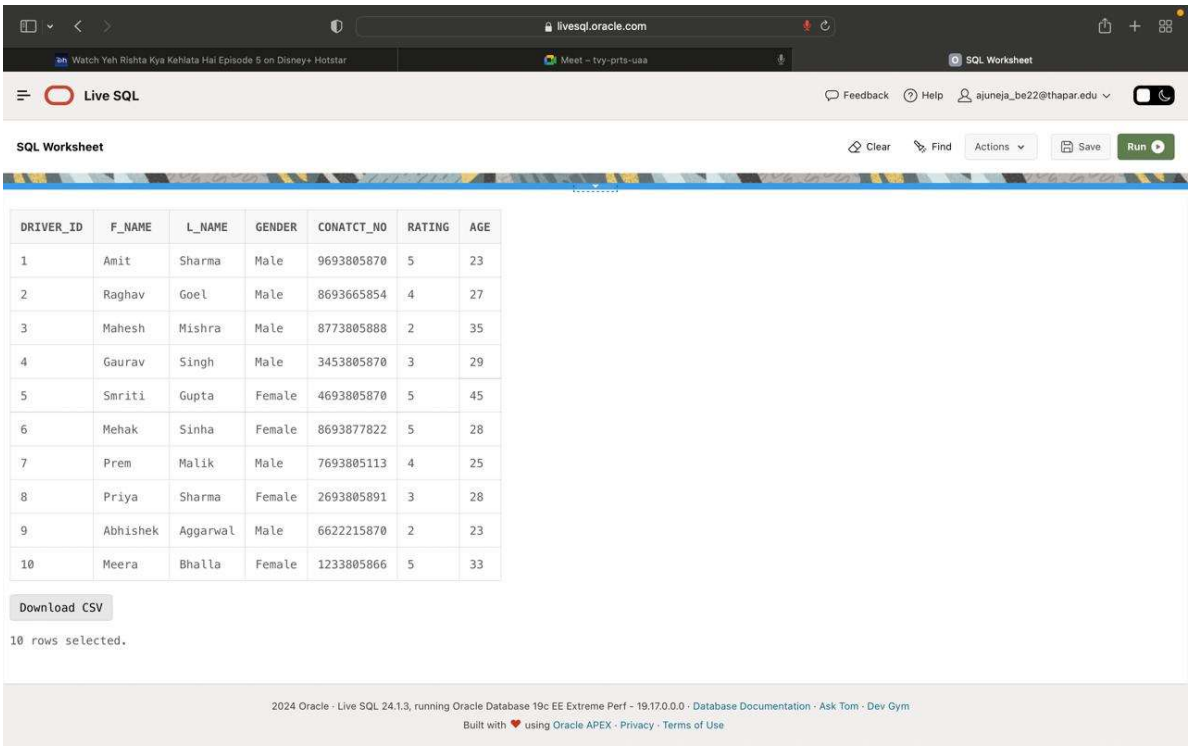


USR_ID	F_NAME	L_NAME	CONTAT_NO	GENDER	ADDRESS	TAXI_ID
201	Raghav	Nayak	9451277009	Male	Bengaluru	105
202	Aryan	Sinha	9876543212	Male	Gurgaon	107
203	Ananya	Pathak	8761122345	Female	Delhi	109
204	Devansh	Merchant	7896544490	Male	Jalandhar	109
205	Akash	Mani	8866459012	Male	Patiala	112
206	Seema	Goel	6578990123	Female	Amritsar	101
207	Vivek	Diwan	7658994321	Male	Dehradun	112
208	Akash	Jha	9866459012	Male	Mumbai	110
209	Pragya	Sharma	9800678443	Female	Chennai	104
210	Aneesha	Sachdev	9993345678	Female	Varanasi	105

CREATION AND INSERTION OF DRIVER

```
CREATE TABLE   DRIVER (
  Driver_id integer NOT NULL,
  F_name VARCHAR(10),
  L_name VARCHAR(20),
  Gender VARCHAR(10),
  Conatct_no VARCHAR(20),
  Rating integer,
  Age integer,
  PRIMARY KEY (Driver_id) );
```

```
INSERT INTO DRIVER VALUES(1,'Amit','Sharma','Male','9693805870',5,23);
INSERT INTO DRIVER VALUES(2,'Raghav','Goel','Male','8693665854',4,27);
INSERT INTO DRIVER VALUES(3,'Mahesh','Mishra','Male','8773805888',2,35);
INSERT INTO DRIVER VALUES(4,'Gaurav','Singh','Male','3453805870',3,29);
INSERT INTO DRIVER VALUES(5,'Smriti','Gupta','Female','4693805870',5,45);
INSERT INTO DRIVER VALUES(6,'Mehak','Sinha','Female','8693877822',5,28);
INSERT INTO DRIVER VALUES(7,'Prem','Malik','Male','7693805113',4,25);
INSERT INTO DRIVER VALUES(8,'Priya','Sharma','Female','2693805891',3,28);
INSERT INTO DRIVER VALUES(9,'Abhishek','Aggarwal','Male','6622215870',2,23);
INSERT INTO DRIVER VALUES(10,'Meera','Bhalla','Female','1233805866',5,33);
```

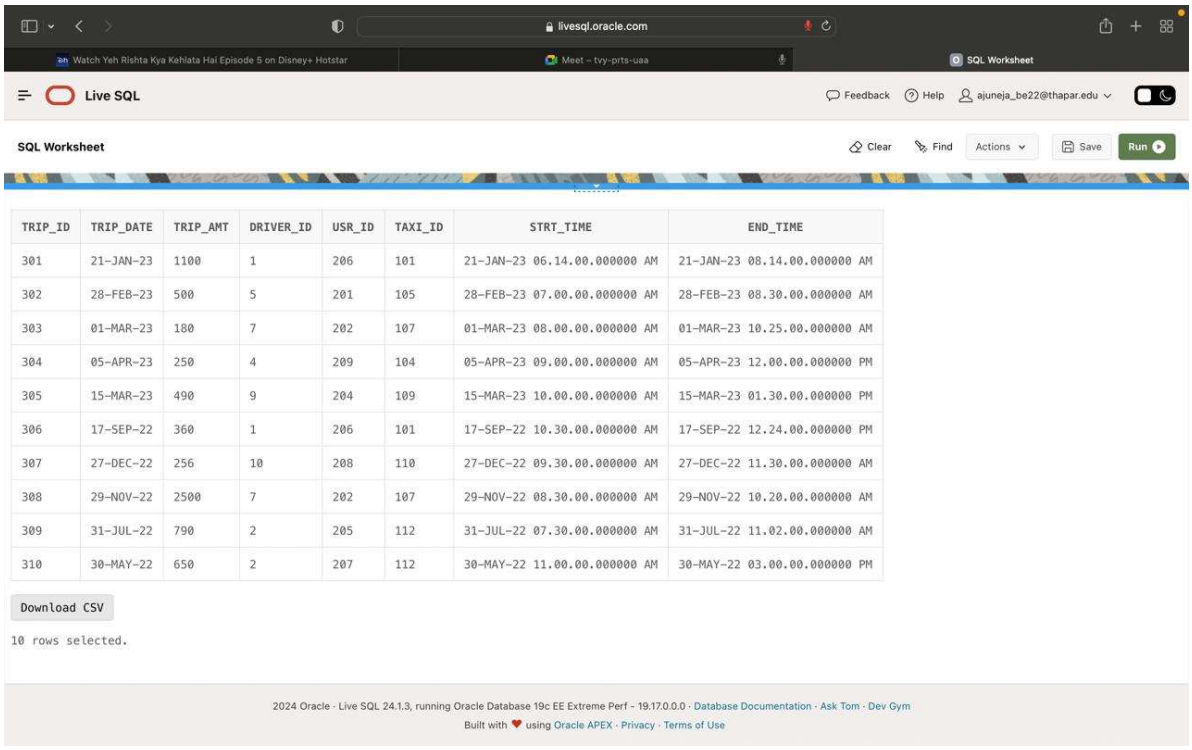


DRIVER_ID	F_NAME	L_NAME	GENDER	CONATCT_NO	RATING	AGE
1	Amit	Sharma	Male	9693805870	5	23
2	Raghav	Goel	Male	8693665854	4	27
3	Mahesh	Mishra	Male	8773805888	2	35
4	Gaurav	Singh	Male	3453805870	3	29
5	Smriti	Gupta	Female	4693805870	5	45
6	Mehak	Sinha	Female	8693877822	5	28
7	Prem	Malik	Male	7693805113	4	25
8	Priya	Sharma	Female	2693805891	3	28
9	Abhishek	Aggarwal	Male	6622215870	2	23
10	Meera	Bhalla	Female	1233805866	5	33

CREATION AND INSERTION OF TRIP_DETAILS

```
CREATE TABLE  TRIP_DETAILS (
  Trip_id integer NOT NULL,
  Trip_date DATE,
  Trip_amt decimal(10,2),
  Driver_id integer,
  Usr_id integer,
  Taxi_id integer,
  Strt_time TIMESTAMP,
```

```
  End_time TIMESTAMP,
  PRIMARY KEY (Trip_id)
);
INSERT INTO TRIP_DETAILS
VALUES(301,to_date('21/01/2023','dd/mm/yyyy'),1100,1,206,101,TO_TIMESTAMP('20
23-01-21 06:14:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2023-01-21
08:14:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(302,to_date('28/02/2023','dd/mm/yyyy'),500,5,201,105,TO_TIMESTAMP('202
3-02-28 07:00:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2023-02-28
08:30:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(303,to_date('01/03/2023','dd/mm/yyyy'),180,7,202,107,TO_TIMESTAMP('202
3-03-01 08:00:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2023-03-01
10:25:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(304,to_date('05/04/2023','dd/mm/yyyy'),250,4,209,104,TO_TIMESTAMP('202
3-04-05 09:00:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2023-04-05
12:00:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(305,to_date('15/03/2023','dd/mm/yyyy'),490,9,204,109,TO_TIMESTAMP('202
3-03-15 10:00:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2023-03-15
13:30:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(306,to_date('17/09/2022','dd/mm/yyyy'),360,1,206,101,TO_TIMESTAMP('202
2-09-17 10:30:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2022-09-17
12:24:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(307,to_date('27/12/2022','dd/mm/yyyy'),256,10,208,110,TO_TIMESTAMP('20
22-12-27 09:30:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2022-12-27
11:30:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(308,to_date('29/11/2022','dd/mm/yyyy'),2500,7,202,107,TO_TIMESTAMP('20
22-11-29 08:30:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2022-11-29
10:20:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(309,to_date('31/07/2022','dd/mm/yyyy'),790,2,205,112,TO_TIMESTAMP('202
2-07-31 07:30:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2022-07-31
11:02:00', 'YYYY-MM-DD HH24:MI:SS'));
INSERT INTO TRIP_DETAILS
VALUES(310,to_date('30/05/2022','dd/mm/yyyy'),650,2,207,112,TO_TIMESTAMP('202
2-05-30 11:00:00', 'YYYY-MM-DD HH24:MI:SS'),TO_TIMESTAMP('2022-05-30
15:00:00', 'YYYY-MM-DD HH24:MI:SS'));
```



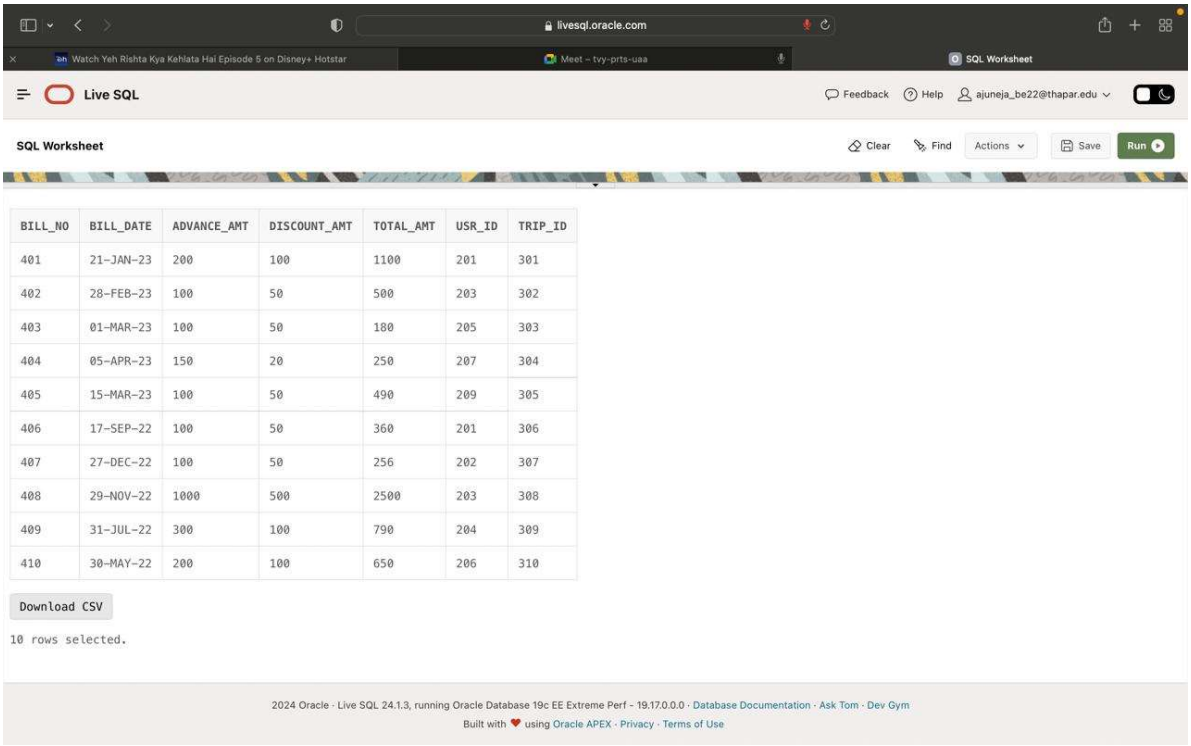
TRIP_ID	TRIP_DATE	TRIP_AMT	DRIVER_ID	USR_ID	TAXI_ID	STRT_TIME	END_TIME
301	21-JAN-23	1100	1	206	101	21-JAN-23 06.14.00.000000 AM	21-JAN-23 08.14.00.000000 AM
302	28-FEB-23	500	5	201	105	28-FEB-23 07.00.00.000000 AM	28-FEB-23 08.30.00.000000 AM
303	01-MAR-23	180	7	202	107	01-MAR-23 08.00.00.000000 AM	01-MAR-23 10.25.00.000000 AM
304	05-APR-23	250	4	209	104	05-APR-23 09.00.00.000000 AM	05-APR-23 12.00.00.000000 PM
305	15-MAR-23	490	9	204	109	15-MAR-23 10.00.00.000000 AM	15-MAR-23 01.30.00.000000 PM
306	17-SEP-22	360	1	206	101	17-SEP-22 10.30.00.000000 AM	17-SEP-22 12.24.00.000000 PM
307	27-DEC-22	256	10	208	110	27-DEC-22 09.30.00.000000 AM	27-DEC-22 11.30.00.000000 AM
308	29-NOV-22	2500	7	202	107	29-NOV-22 08.30.00.000000 AM	29-NOV-22 10.20.00.000000 AM
309	31-JUL-22	790	2	205	112	31-JUL-22 07.30.00.000000 AM	31-JUL-22 11.02.00.000000 AM
310	30-MAY-22	650	2	207	112	30-MAY-22 11.00.00.000000 AM	30-MAY-22 03.00.00.000000 PM

CREATION AND INSERTION OF BILL_DETAILS

```
CREATE TABLE BILL_DETAILS (
  Bill_no integer NOT NULL,
  Bill_date DATE,
  Advance_amt decimal(10,2),
  Discount_amt decimal(10,2),
  Total_amt decimal(10,2),
  Usr_id integer,
  Trip_id integer,
  PRIMARY KEY (Bill_no),
  UNIQUE (Trip_id)
);
```

```
INSERT INTO BILL_DETAILS
VALUES(401,to_date('21/01/2023','dd/mm/yyyy'),200,100,1100,201,301);
INSERT INTO BILL_DETAILS
VALUES(402,to_date('28/02/2023','dd/mm/yyyy'),100,50,500,203,302);
INSERT INTO BILL_DETAILS
VALUES(403,to_date('01/03/2023','dd/mm/yyyy'),100,50,180,205,303);
INSERT INTO BILL_DETAILS
VALUES(404,to_date('05/04/2023','dd/mm/yyyy'),150,20,250,207,304);
```

```
INSERT INTO BILL_DETAILS
VALUES(405,to_date('15/03/2023','dd/mm/yyyy'),100,50,490,209,305);
INSERT INTO BILL_DETAILS
VALUES(406,to_date('17/09/2022','dd/mm/yyyy'),100,50,360,201,306);
INSERT INTO BILL_DETAILS
VALUES(407,to_date('27/12/2022','dd/mm/yyyy'),100,50,256,202,307);
INSERT INTO BILL_DETAILS
VALUES(408,to_date('29/11/2022','dd/mm/yyyy'),1000,500,2500,203,308);
INSERT INTO BILL_DETAILS
VALUES(409,to_date('31/07/2022','dd/mm/yyyy'),300,100,790,204,309);
INSERT INTO BILL_DETAILS
VALUES(410,to_date('30/05/2022','dd/mm/yyyy'),200,100,650,206,310);
```

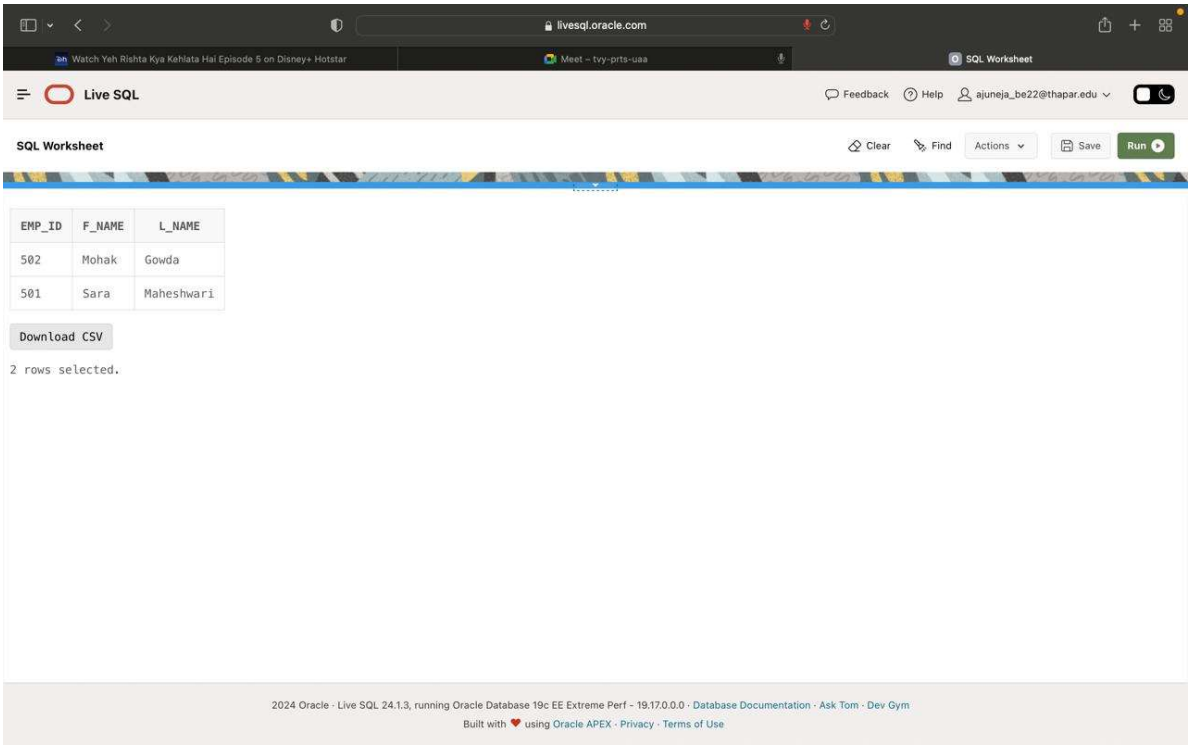


BILL_NO	BILL_DATE	ADVANCE_AMT	DISCOUNT_AMT	TOTAL_AMT	USR_ID	TRIP_ID
401	21-JAN-23	200	100	1100	201	301
402	28-FEB-23	100	50	500	203	302
403	01-MAR-23	100	50	180	205	303
404	05-APR-23	150	20	250	207	304
405	15-MAR-23	100	50	490	209	305
406	17-SEP-22	100	50	360	201	306
407	27-DEC-22	100	50	256	202	307
408	29-NOV-22	1000	500	2500	203	308
409	31-JUL-22	300	100	790	204	309
410	30-MAY-22	200	100	650	206	310

CREATION AND INSERTION OF CUSTOMER_SERVICE

```
CREATE TABLE  CUSTOMER_SERVICE (
  Emp_id integer NOT NULL,
  F_name VARCHAR(20),
  L_name VARCHAR(20),
  PRIMARY KEY (Emp_id)
);
```

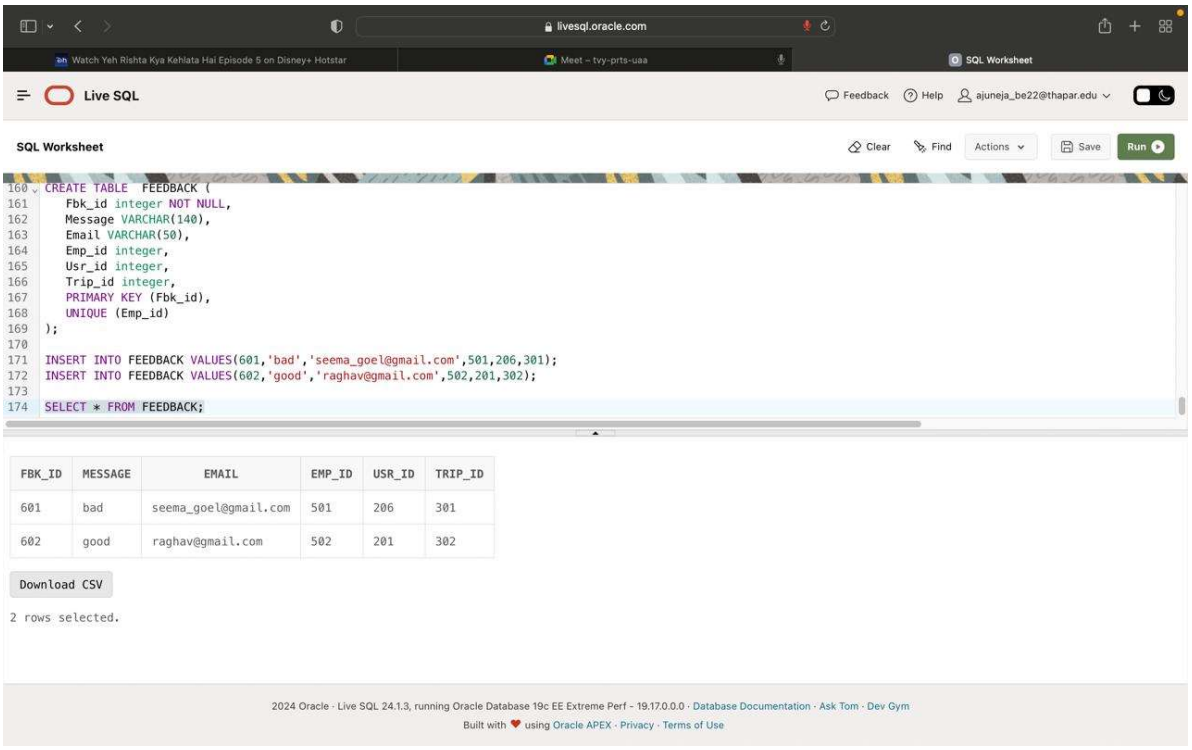
NSERT INTO CUSTOMER_SERVICE VALUES(501,'Sara','Maheshwari');
INSERT INTO CUSTOMER_SERVICE VALUES(502,'Mohak','Gowda');



CREATION AND INSERTION OF FEEDBACK

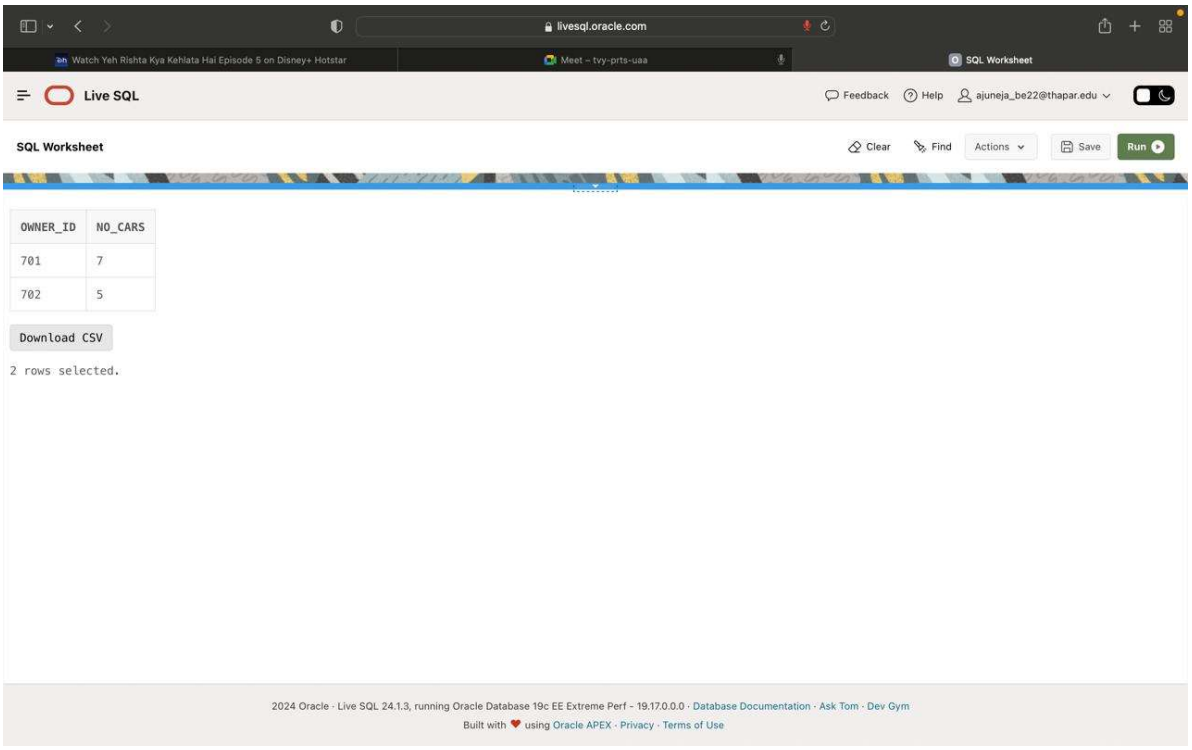
```
CREATE TABLE  FEEDBACK (
  Fbk_id integer NOT NULL,
  Message VARCHAR(140),
  Email VARCHAR(50),
  Emp_id integer,
  Usr_id integer,
  Trip_id integer,
  PRIMARY KEY (Fbk_id),
  UNIQUE (Emp_id)
);
```

INSERT INTO FEEDBACK VALUES(601,'bad','seema_goel@gmail.com',501,206,301);
INSERT INTO FEEDBACK VALUES(602,'good','raghav@gmail.com',502,201,302);



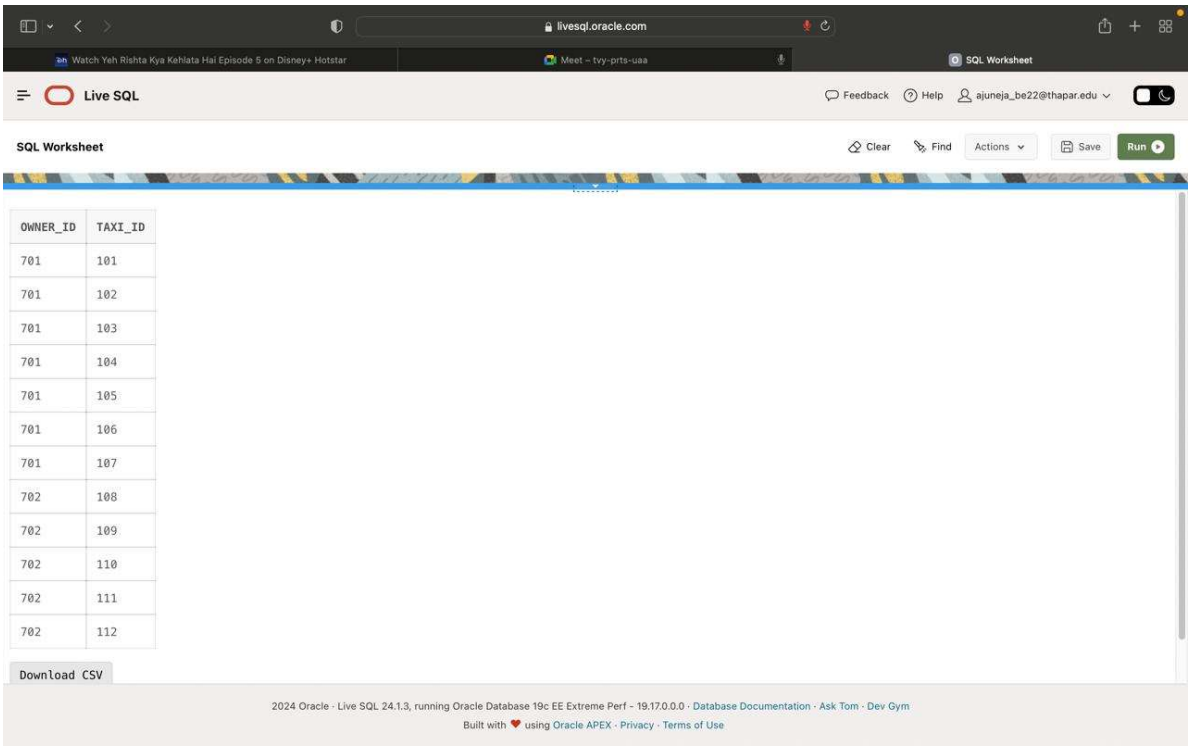
CREATION AND INSERTION OF OWNS

```
CREATE TABLE  OWNS (
  Owner_id integer NOT NULL,
  No_Cars  integer,
  PRIMARY KEY (Owner_id)
);

INSERT INTO OWNS VALUES(701,7);
INSERT INTO OWNS VALUES(702,5);
```



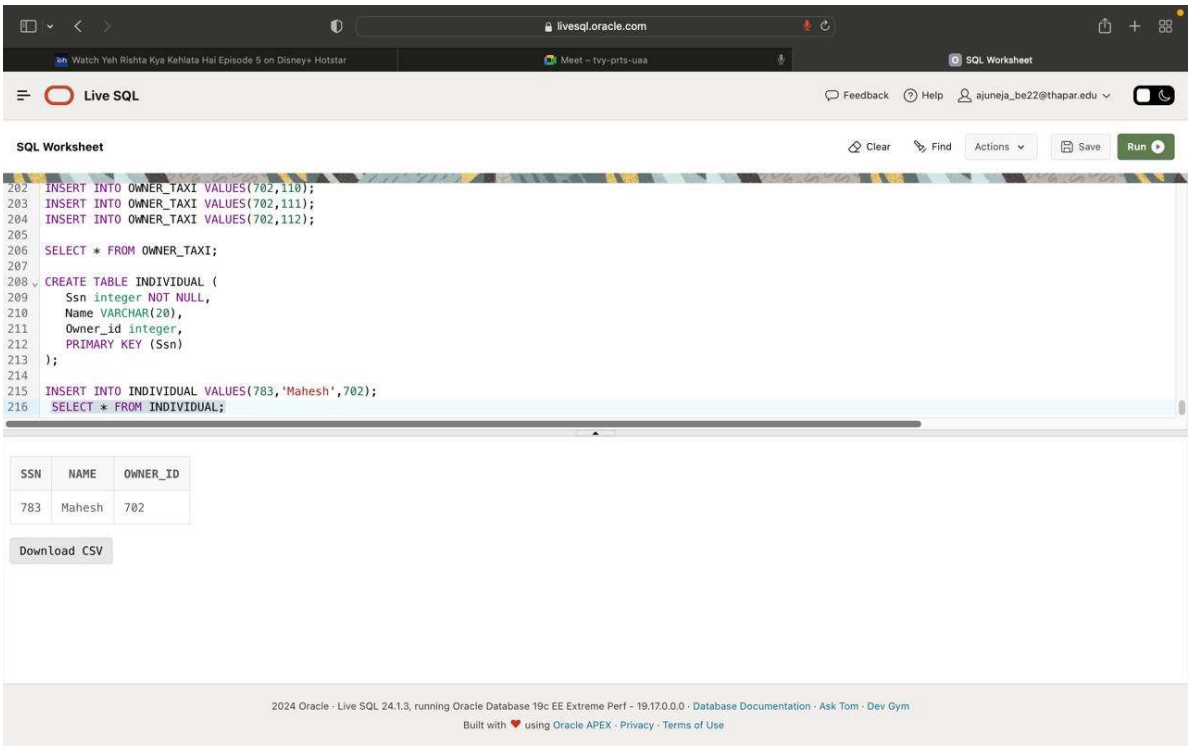
CREATION AND INSERTION OF OWNER_TAXI

```
CREATE TABLE  OWNER_TAXI (
  Owner_id integer NOT NULL,
  Taxi_id integer,
  PRIMARY KEY (Owner_id, Taxi_id)
);

INSERT INTO OWNER_TAXI VALUES(701,101);
INSERT INTO OWNER_TAXI VALUES(701,102);
INSERT INTO OWNER_TAXI VALUES(701,103);
INSERT INTO OWNER_TAXI VALUES(701,104);
INSERT INTO OWNER_TAXI VALUES(701,105);
INSERT INTO OWNER_TAXI VALUES(701,106);
INSERT INTO OWNER_TAXI VALUES(701,107);
INSERT INTO OWNER_TAXI VALUES(702,108);
INSERT INTO OWNER_TAXI VALUES(702,109);
INSERT INTO OWNER_TAXI VALUES(702,110);
INSERT INTO OWNER_TAXI VALUES(702,111);
INSERT INTO OWNER_TAXI VALUES(702,112);
```



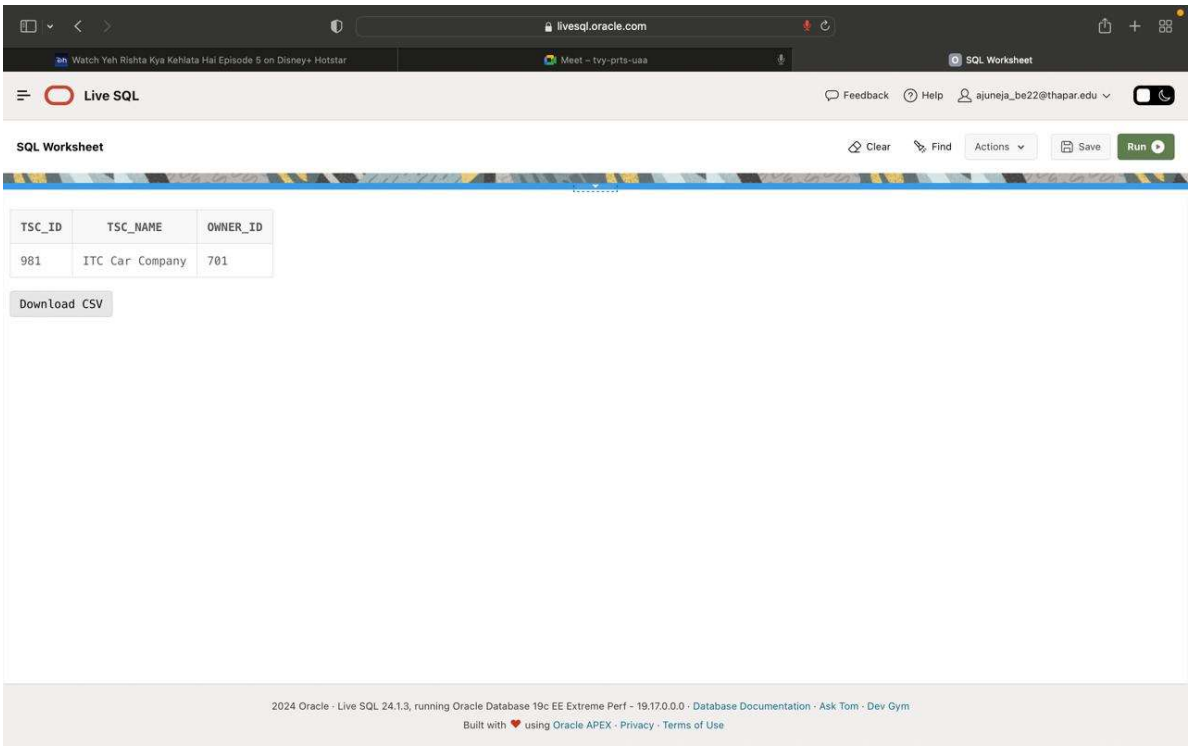
CREATION AND INSERTION OF INDIVIDUAL

```
CREATE TABLE INDIVIDUAL (
  Ssn integer NOT NULL,
  Name VARCHAR(20),
  Owner_id integer,
  PRIMARY KEY (Ssn)
);

INSERT INTO INDIVIDUAL VALUES(783,'Mahesh',702);
```



CREATION AND INSERTION OF TAXI_SERVICE_COMPANY

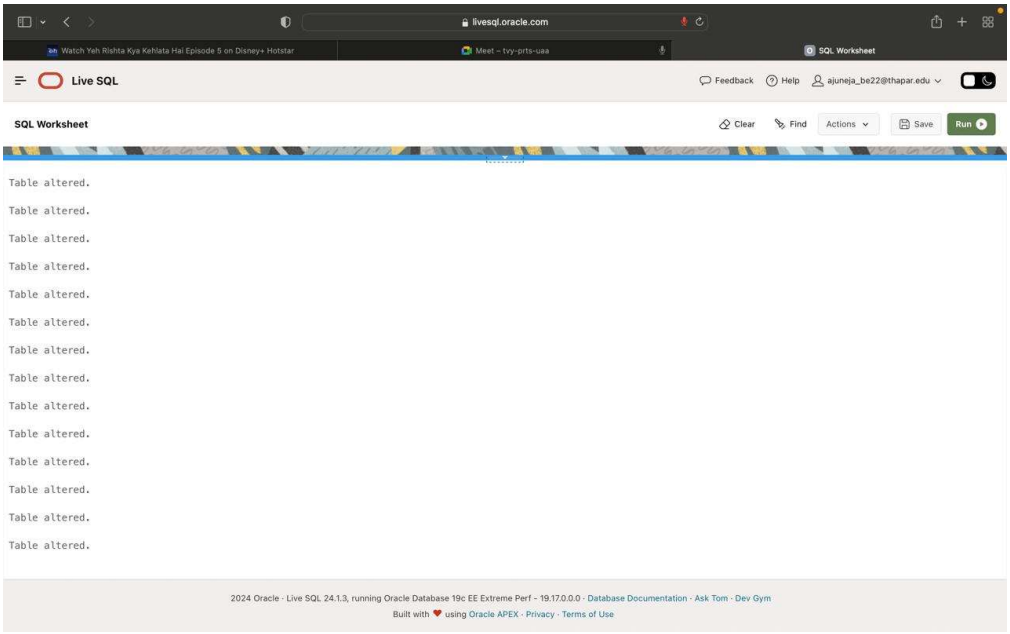
```
CREATE TABLE  TAXI_SERVICE_COMPANY (
  Tsc_id integer NOT NULL,
  Tsc_name VARCHAR(20),
  Owner_id integer,
  PRIMARY KEY (Tsc_id)
);

INSERT INTO TAXI_SERVICE_COMPANY VALUES (981,'ITC Car Company',701);
```



B) CONSTRAINTS

```
ALTER TABLE TAXI ADD CONSTRAINT fk1 FOREIGN KEY (Driver_id)
REFERENCES DRIVER(Driver_id) ON DELETE CASCADE;
ALTER TABLE USER_TBL ADD CONSTRAINT fk2 FOREIGN KEY (Taxi_id)
REFERENCES TAXI(Taxi_id) ON DELETE CASCADE;
ALTER TABLE TRIP_DETAILS ADD CONSTRAINT fk3 FOREIGN KEY (Driver_id)
REFERENCES DRIVER(Driver_id) ON DELETE CASCADE;
ALTER TABLE TRIP_DETAILS ADD CONSTRAINT fk4 FOREIGN KEY (Usr_id)
REFERENCES USER_TBL(Usr_id) ON DELETE CASCADE;
ALTER TABLE TRIP_DETAILS ADD CONSTRAINT fk5 FOREIGN KEY (Taxi_id)
REFERENCES TAXI(Taxi_id) ON DELETE CASCADE;
ALTER TABLE BILL_DETAILS ADD CONSTRAINT fk6 FOREIGN KEY (Trip_id)
REFERENCES TRIP_DETAILS(Trip_id) ON DELETE CASCADE;
ALTER TABLE BILL_DETAILS ADD CONSTRAINT fk7 FOREIGN KEY (Usr_id)
REFERENCES USER_TBL(Usr_id) ON DELETE CASCADE;
ALTER TABLE FEEDBACK ADD CONSTRAINT fk8 FOREIGN KEY (Emp_id)
REFERENCES CUSTOMER_SERVICE(Emp_id) ON DELETE CASCADE;
ALTER TABLE FEEDBACK ADD CONSTRAINT fk9 FOREIGN KEY (Trip_id)
REFERENCES TRIP_DETAILS(Trip_id) ON DELETE CASCADE;
ALTER TABLE FEEDBACK ADD CONSTRAINT fk10 FOREIGN KEY (Usr_id)
REFERENCES USER_TBL(Usr_id) ON DELETE CASCADE;
```

```
ALTER TABLE OWNER_TAXI ADD CONSTRAINT fk11 FOREIGN KEY (Taxi_id)
REFERENCES TAXI(Taxi_id) ON DELETE CASCADE;
ALTER TABLE OWNER_TAXI ADD CONSTRAINT fk12 FOREIGN KEY (Owner_id)
REFERENCES OWNS(Owner_id) ON DELETE CASCADE;
ALTER TABLE INDIVIDUAL ADD CONSTRAINT fk13 FOREIGN KEY (Owner_id)
REFERENCES OWNS(Owner_id) ON DELETE CASCADE;
ALTER TABLE TAXI_SERVICE_COMPANY ADD CONSTRAINT fk14 FOREIGN
KEY (Owner_id) REFERENCES OWNS(Owner_id) ON DELETE CASCADE;
```

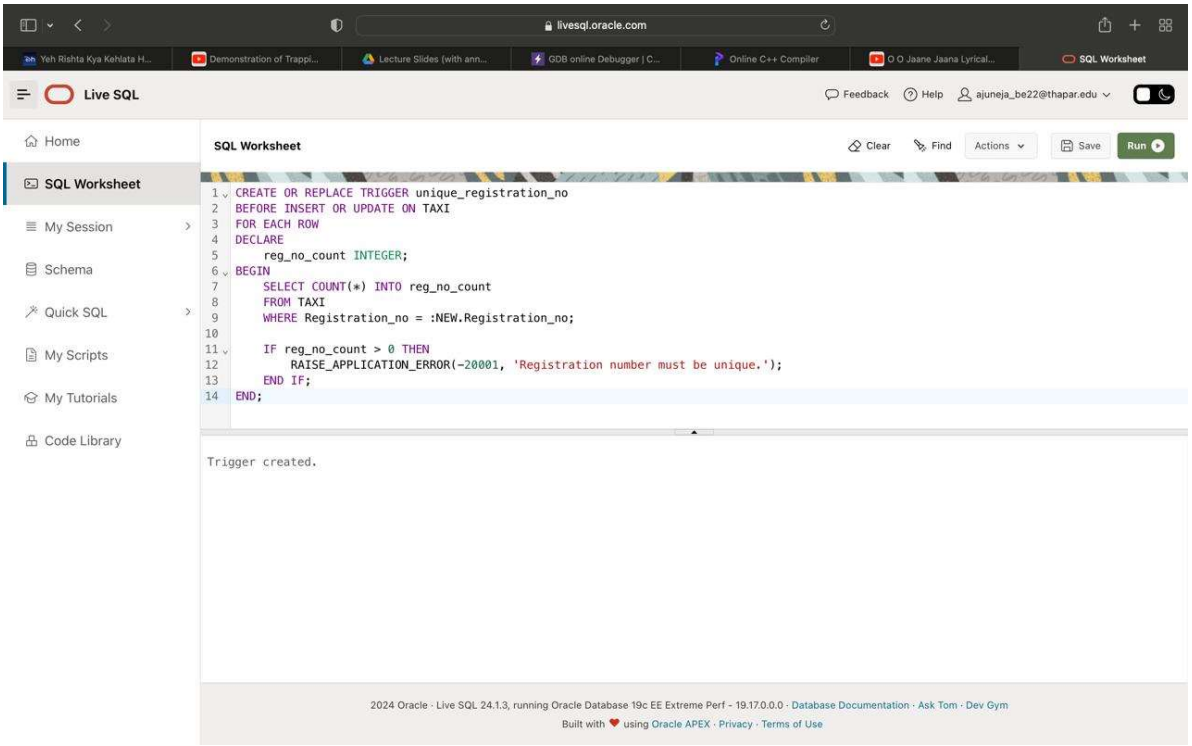


C) TRIGGERS

TRIGGER FOR ENSURING UNIQUE REGISTRATION NUMBER FOR TAXI TABLE

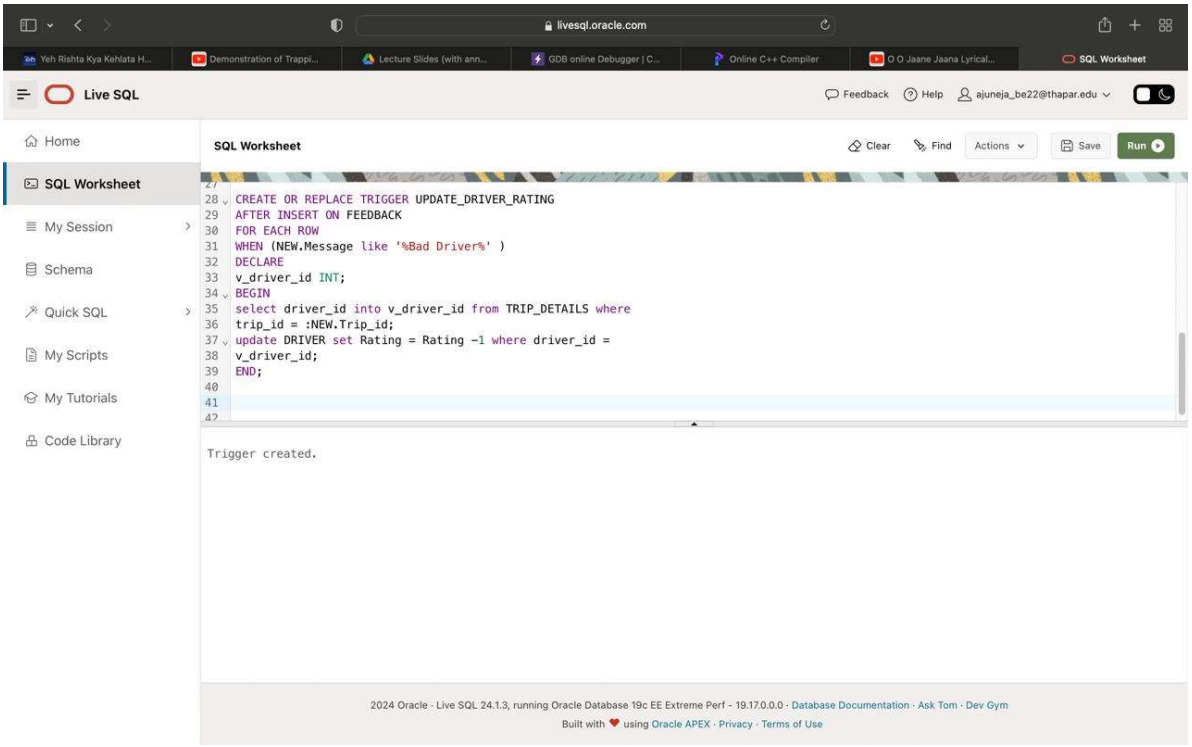
```
CREATE OR REPLACE TRIGGER unique_registration_no
BEFORE INSERT OR UPDATE ON TAXI
FOR EACH ROW
DECLARE
   reg_no_count INTEGER;
BEGIN
   SELECT COUNT(*) INTO reg_no_count
   FROM TAXI
   WHERE Registration_no = :NEW.Registration_no;
```

```
    IF reg_no_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Registration number must be unique.');
    END IF;
END;
```



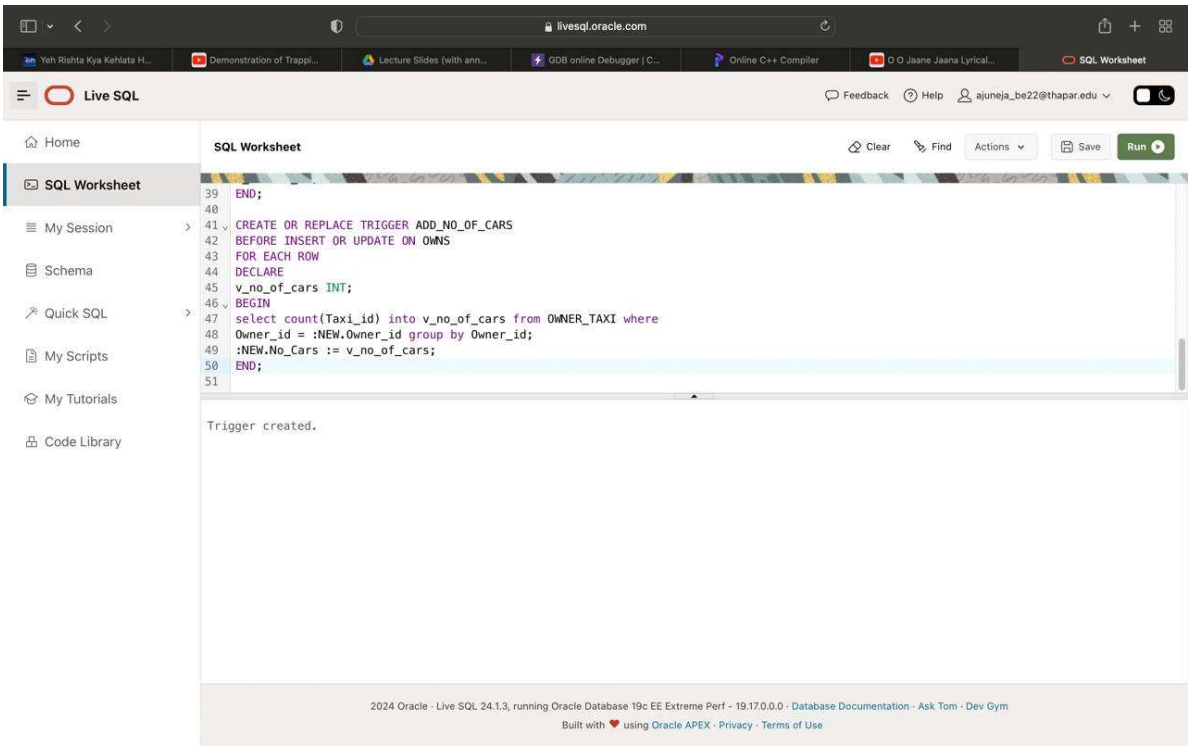
TRIGGER TO DECREASE THE DRIVER RATING BY 1 IF USER FEEDBACK IS BAD FOR A DRIVER FOR FEEDBACK TABLE

```
CREATE OR REPLACE TRIGGER UPDATE_DRIVER_RATING
AFTER INSERT ON FEEDBACK
FOR EACH ROW
WHEN (NEW.Message like '%Bad Driver%' )
DECLARE
v_driver_id INT;
BEGIN
select driver_id into v_driver_id from TRIP_DETAILS where
trip_id = :NEW.Trip_id;
update DRIVER set Rating = Rating -1 where driver_id =
v_driver_id;
END;
```



TRIGGER TO CALCULATE THE NUMBER OF CARS OWNED BY THE OWNER AND UPDATE THE NO_OF_CARS COLUMNS IN THE OWNS TABLE

```
CREATE OR REPLACE TRIGGER ADD_NO_OF_CARS
BEFORE INSERT OR UPDATE ON OWNS
FOR EACH ROW
DECLARE
v_no_of_cars INT;
BEGIN
select count(Taxi_id) into v_no_of_cars from OWNER_TAXI where
Owner_id = :NEW.Owner_id group by Owner_id;
:NEW.No_Cars := v_no_of_cars;
END;
```

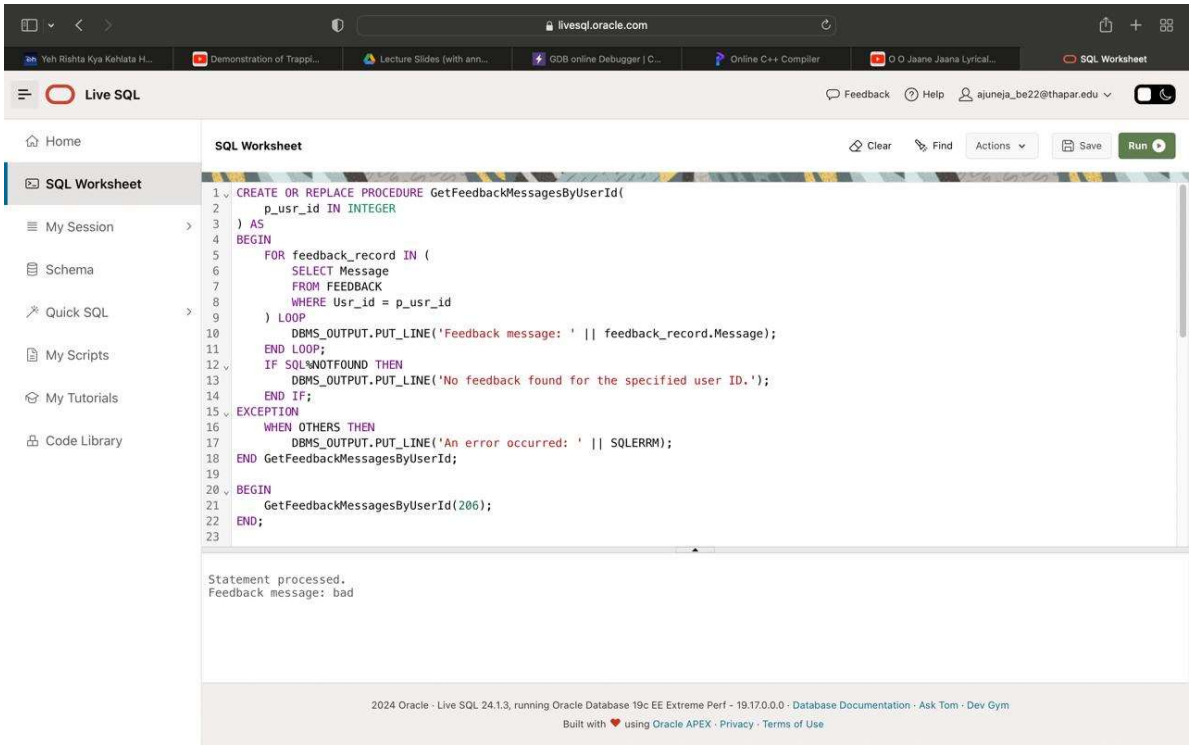


D) PROCEDURES

PROCEDURE TO FETCH THE FEEDBACK MESSAGES FOR THE SPECIFIED USER ID

```
CREATE OR REPLACE PROCEDURE GetFeedbackMessagesByUserId(
   p_usr_id IN INTEGER
) AS
BEGIN
   FOR feedback_record IN (
      SELECT Message
      FROM FEEDBACK
      WHERE Usr_id = p_usr_id
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Feedback message: ' || feedback_record.Message);
   END LOOP;
   IF SQL%NOTFOUND THEN
      DBMS_OUTPUT.PUT_LINE('No feedback found for the specified user ID.');
   END IF;
EXCEPTION
```

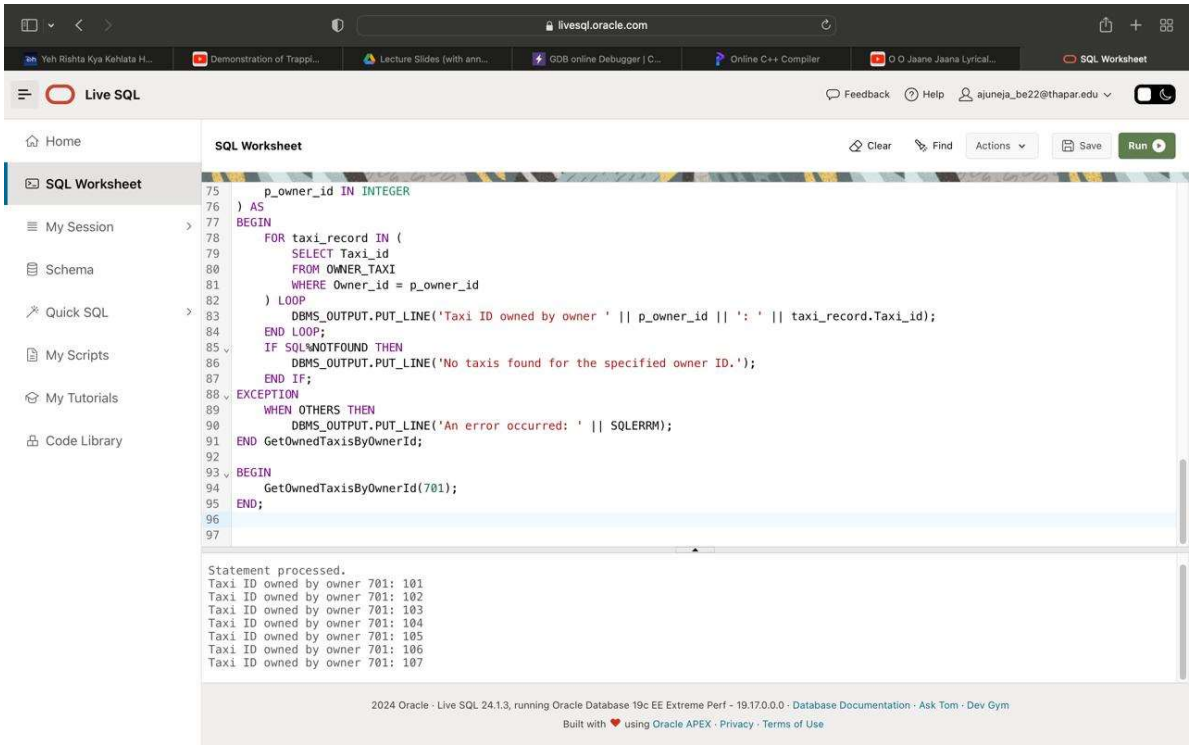
```
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END GetFeedbackMessagesByUserId;

BEGIN
    GetFeedbackMessagesByUserId(206);
END;
```



PROCEDURE TO RETRIEVE OWNED TAXIS FOR DIFFERENT OWNER IDS

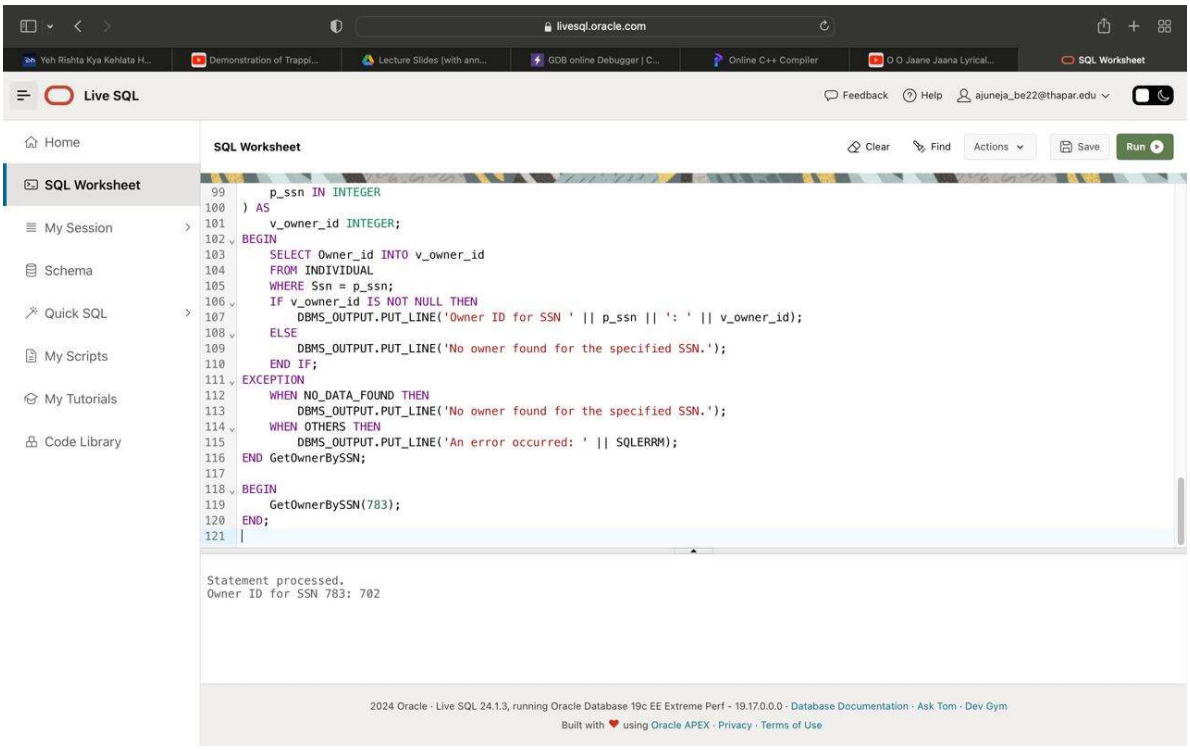
```
CREATE OR REPLACE PROCEDURE GetOwnedTaxisByOwnerId(
    p_owner_id IN INTEGER
) AS
BEGIN
    FOR taxi_record IN (
        SELECT Taxi_id
        FROM OWNER_TAXI
        WHERE Owner_id = p_owner_id
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Taxi ID owned by owner ' || p_owner_id || ': ' ||
taxi_record.Taxi_id);
```

```
    END LOOP;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No taxis found for the specified owner ID.');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END GetOwnedTaxisByOwnerId;


BEGIN
    GetOwnedTaxisByOwnerId(701);
END;
```



PROCEDURE TO RETRIEVE AND DISPLAY THE OWNER ID FOR SPECIFIED SSN

```
CREATE OR REPLACE PROCEDURE GetOwnerBySSN(
   p_ssn IN INTEGER
) AS
   v_owner_id INTEGER;
BEGIN
   SELECT Owner_id INTO v_owner_id
   FROM INDIVIDUAL
   WHERE Ssn = p_ssn;
   IF v_owner_id IS NOT NULL THEN
      DBMS_OUTPUT.PUT_LINE('Owner ID for SSN ' || p_ssn || ': ' || v_owner_id);
   ELSE
      DBMS_OUTPUT.PUT_LINE('No owner found for the specified SSN.');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No owner found for the specified SSN.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END GetOwnerBySSN;

BEGIN
   GetOwnerBySSN(783);
END;
```

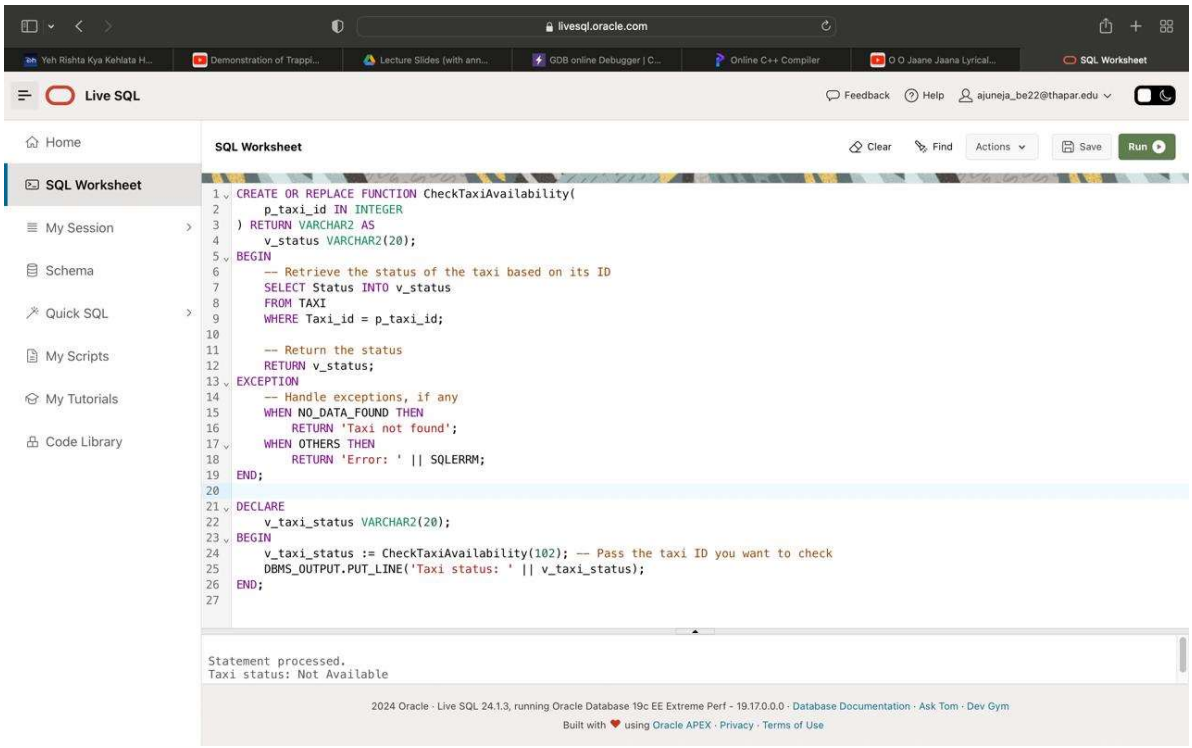


E) FUNCTIONS

FUNCTION TO CHECK THE AVAILABILITY OF TAXI

```
CREATE OR REPLACE FUNCTION CheckTaxiAvailability(
   p_taxi_id IN INTEGER
) RETURN VARCHAR2 AS
   v_status VARCHAR2(20);
BEGIN
   SELECT Status INTO v_status
   FROM TAXI
   WHERE Taxi_id = p_taxi_id;
   RETURN v_status;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 'Taxi not found';
   WHEN OTHERS THEN
      RETURN 'Error: ' || SQLERRM;
END;
DECLARE
   v_taxi_status VARCHAR2(20);
BEGIN
```

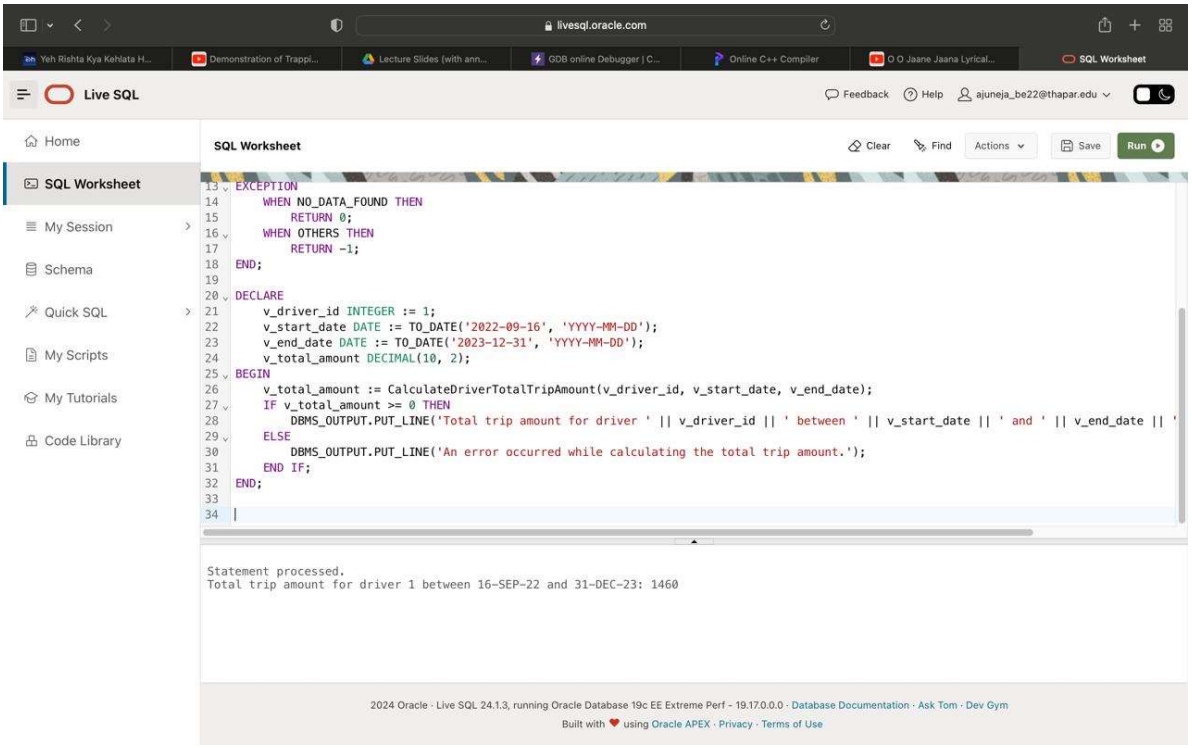
```
    v_taxi_status := CheckTaxiAvailability(102);
    DBMS_OUTPUT.PUT_LINE('Taxi status: ' || v_taxi_status);
END;
```



FUNCTION TO CALCULATE THE TOTAL TRIP AMOUNT FOR A GIVEN DRIVER WITHIN A SPECIFIED DATE RANGE

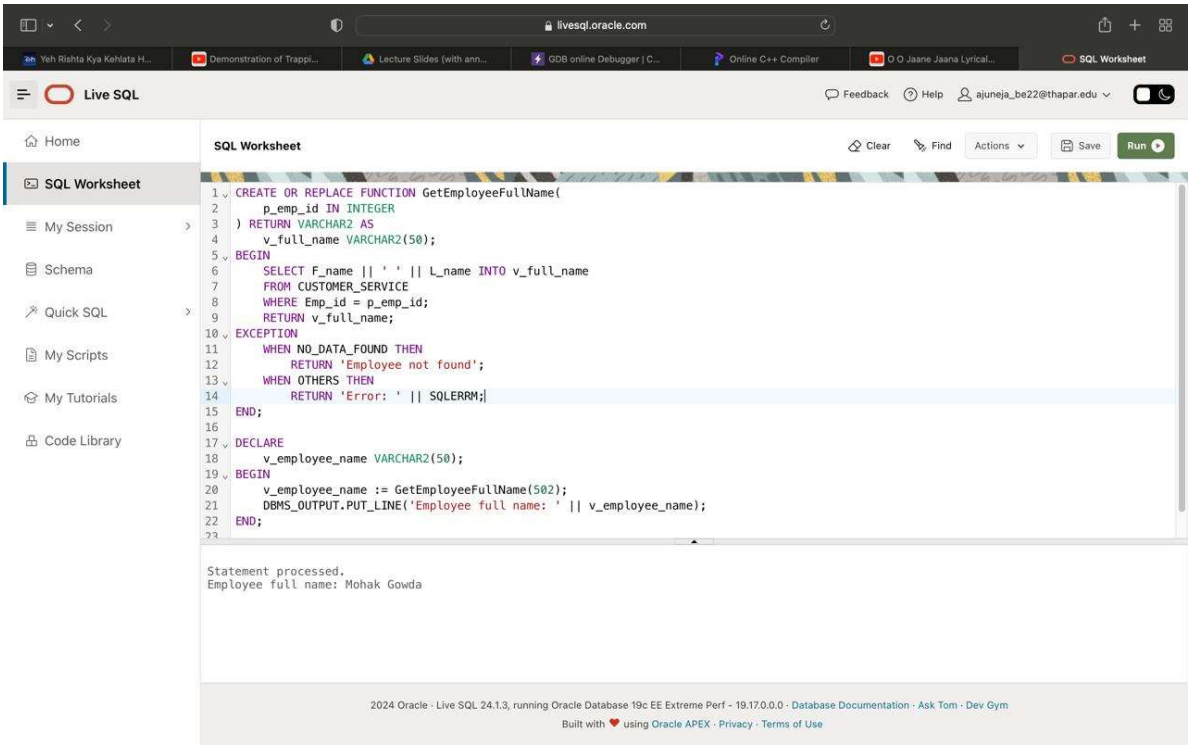
```
CREATE OR REPLACE FUNCTION CalculateDriverTotalTripAmount(
    p_driver_id IN INTEGER,
    p_start_date IN DATE,
    p_end_date IN DATE
) RETURN DECIMAL AS
    v_total_trip_amount DECIMAL(10, 2) :=0;
BEGIN
    SELECT SUM(Trip_amt) INTO v_total_trip_amount
    FROM TRIP_DETAILS
    WHERE Driver_id = p_driver_id
    AND Trip_date BETWEEN p_start_date AND p_end_date;
    RETURN v_total_trip_amount;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
    WHEN OTHERS THEN
```

```
        RETURN -1;
END;

DECLARE
   v_driver_id INTEGER := 1;
   v_start_date DATE := TO_DATE('2022-09-16', 'YYYY-MM-DD');
   v_end_date DATE := TO_DATE('2023-12-31', 'YYYY-MM-DD');
   v_total_amount DECIMAL(10, 2);
BEGIN
   v_total_amount := CalculateDriverTotalTripAmount(v_driver_id, v_start_date,
v_end_date);
   IF v_total_amount >= 0 THEN
      DBMS_OUTPUT.PUT_LINE('Total trip amount for driver ' || v_driver_id || '
between ' || v_start_date || ' and ' || v_end_date || ': ' || v_total_amount);
   ELSE
      DBMS_OUTPUT.PUT_LINE('An error occurred while calculating the total trip
amount.');
   END IF;
END;
```



FUNCTION TO RETRIEVE THE FULL NAME OF THE EMPLOYEE WHEN THE EMPLOYEE ID IS GIVEN

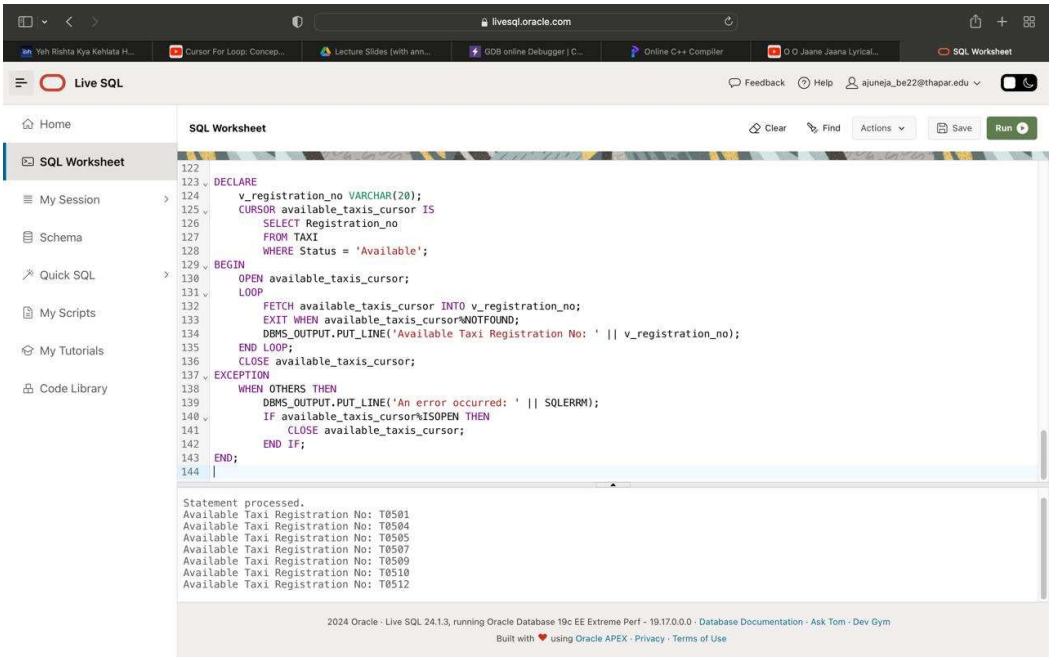
```
CREATE OR REPLACE FUNCTION GetEmployeeFullName(
   p_emp_id IN INTEGER
) RETURN VARCHAR2 AS
   v_full_name VARCHAR2(50);
BEGIN
   SELECT F_name || ' ' || L_name INTO v_full_name
   FROM CUSTOMER_SERVICE
   WHERE Emp_id = p_emp_id;
   RETURN v_full_name;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 'Employee not found';
   WHEN OTHERS THEN
      RETURN 'Error: ' || SQLERRM;
END;

DECLARE
   v_employee_name VARCHAR2(50);
BEGIN
   v_employee_name := GetEmployeeFullName(502);
   DBMS_OUTPUT.PUT_LINE('Employee full name: ' || v_employee_name);
END;
```



F) CURSORS

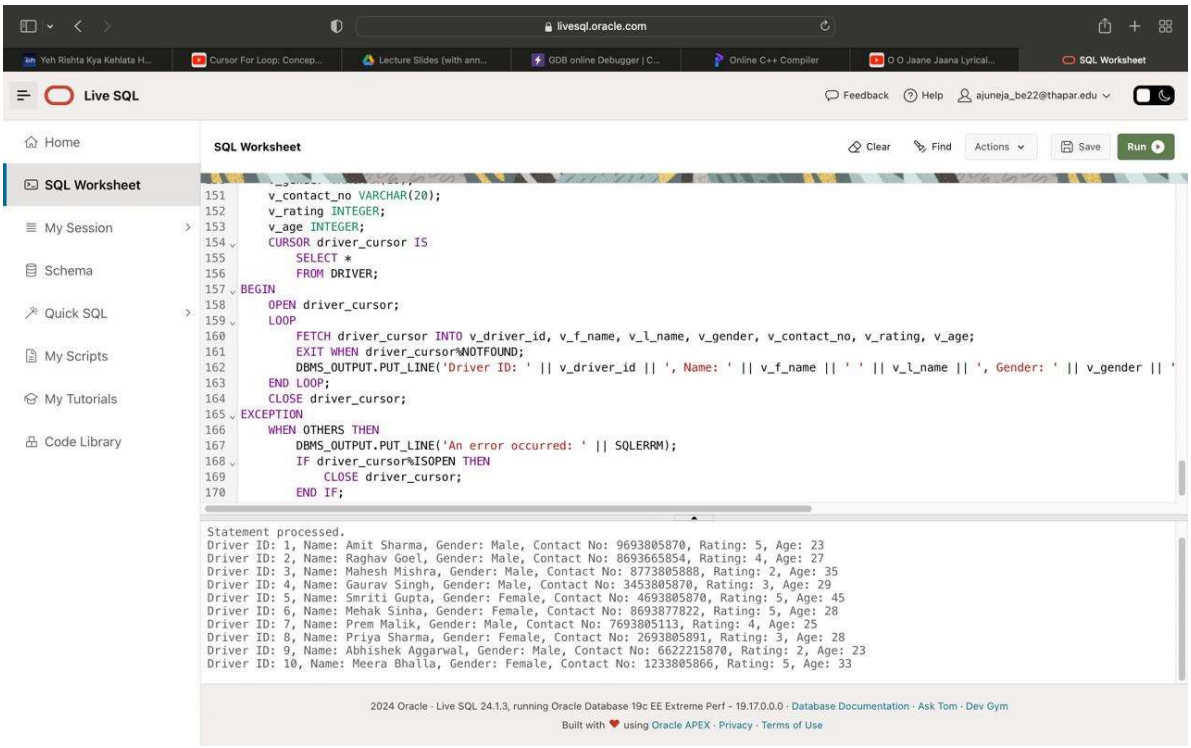
CURSOR TO FETCH EACH REGISTRATION NUMBER ONE BY ONE, AND DISPLAY IT FOR ‘AVAILABLE’ TAXIS

```
DECLARE
   v_registration_no VARCHAR(20);
   CURSOR available_taxis_cursor IS
      SELECT Registration_no
      FROM TAXI
      WHERE Status = 'Available';
BEGIN
   OPEN available_taxis_cursor;
   LOOP
      FETCH available_taxis_cursor INTO v_registration_no;
      EXIT WHEN available_taxis_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Available Taxi Registration No: ' || v_registration_no);
   END LOOP;
   CLOSE available_taxis_cursor;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
      IF available_taxis_cursor%ISOPEN THEN
         CLOSE available_taxis_cursor;
      END IF;
END;
```



CURSOR TO RETRIEVE ALL COLUMNS FROM THE DRIVER TABLE AND DISPLAYS THE DETAILS OF EACH DRIVER

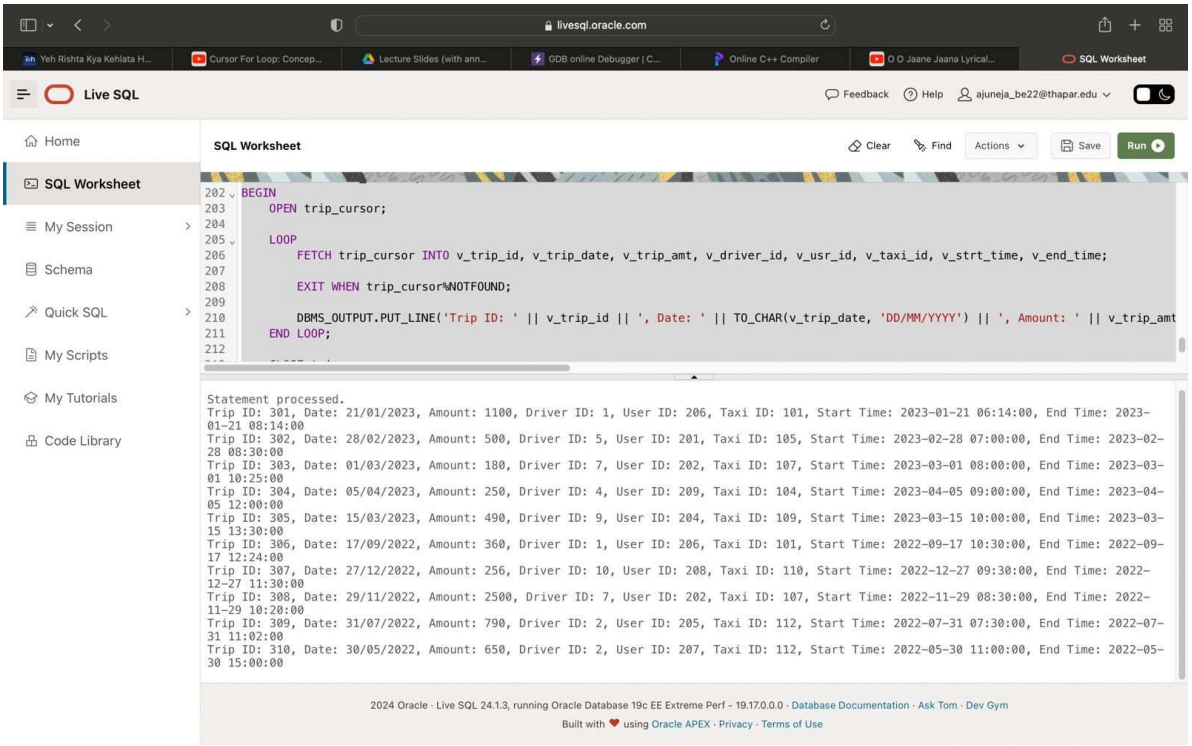
```
DECLARE
   v_driver_id INTEGER;
   v_f_name VARCHAR(10);
   v_l_name VARCHAR(20);
   v_gender VARCHAR(10);
   v_contact_no VARCHAR(20);
   v_rating INTEGER;
   v_age INTEGER;
   CURSOR driver_cursor IS
      SELECT *
      FROM DRIVER;
BEGIN
   OPEN driver_cursor;
   LOOP
      FETCH driver_cursor INTO v_driver_id, v_f_name, v_l_name, v_gender,
v_contact_no, v_rating, v_age;
      EXIT WHEN driver_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Driver ID: ' || v_driver_id || ', Name: ' || v_f_name || ' '
|| v_l_name || ', Gender: ' || v_gender || ', Contact No: ' || v_contact_no || ', Rating: ' ||
v_rating || ', Age: ' || v_age);
   END LOOP;
   CLOSE driver_cursor;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
      IF driver_cursor%ISOPEN THEN
         CLOSE driver_cursor;
      END IF;
END;
```



CURSOR TO DISPLAY THE DETAILS OF EACH TRIP FROM TRIP_DETAILS

```
DECLARE
   v_trip_id INTEGER;
   v_trip_date DATE;
   v_trip_amt DECIMAL(10, 2);
   v_driver_id INTEGER;
   v_usr_id INTEGER;
   v_taxi_id INTEGER;
   v_strt_time TIMESTAMP;
   v_end_time TIMESTAMP;

   CURSOR trip_cursor IS
      SELECT *
      FROM TRIP_DETAILS;
BEGIN
   OPEN trip_cursor;

   LOOP
      FETCH trip_cursor INTO v_trip_id, v_trip_date, v_trip_amt, v_driver_id, v_usr_id,
v_taxi_id, v_strt_time, v_end_time;

      EXIT WHEN trip_cursor%NOTFOUND;
```

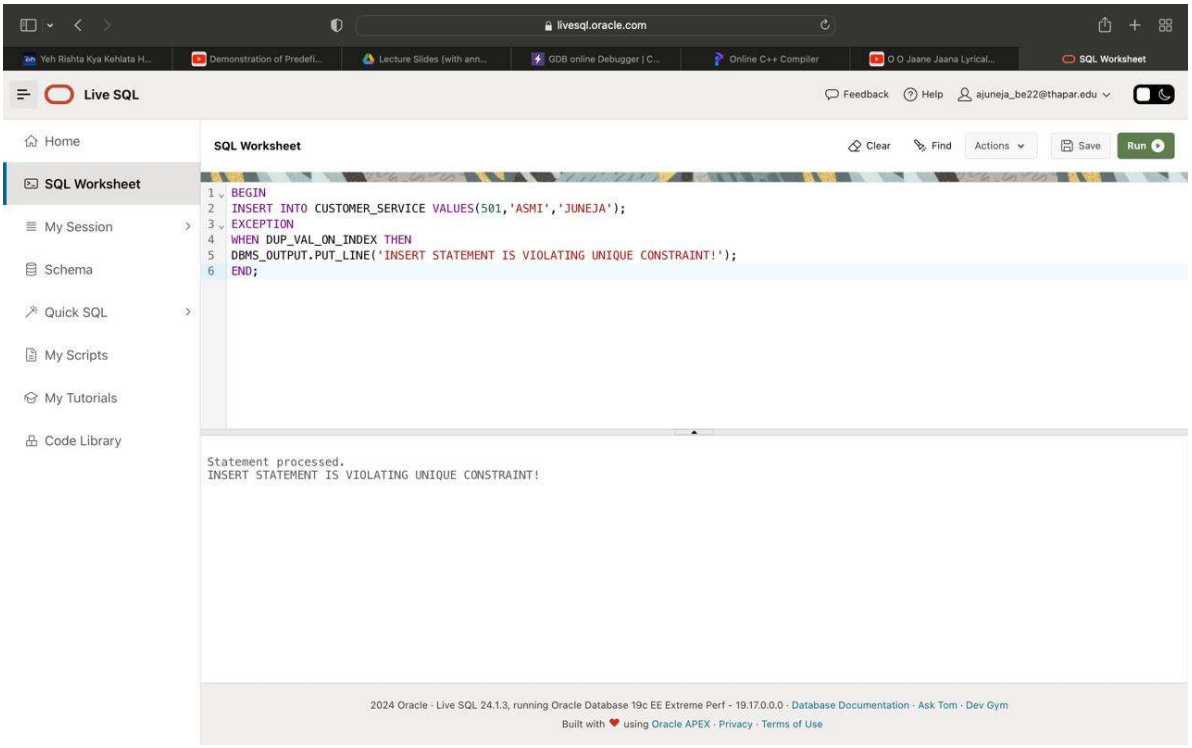
```
        DBMS_OUTPUT.PUT_LINE('Trip ID: ' || v_trip_id || ', Date: ' ||
TO_CHAR(v_trip_date, 'DD/MM/YYYY') || ', Amount: ' || v_trip_amt || ', Driver ID: ' ||
v_driver_id || ', User ID: ' || v_usr_id || ', Taxi ID: ' || v_taxi_id || ', Start Time: ' ||
TO_CHAR(v_strt_time, 'YYYY-MM-DD HH24:MI:SS') || ', End Time: ' ||
TO_CHAR(v_end_time, 'YYYY-MM-DD HH24:MI:SS'));
    END LOOP;

    CLOSE trip_cursor;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        IF trip_cursor%ISOPEN THEN
            CLOSE trip_cursor;
        END IF;
END;
```



G)EXCEPTIONS

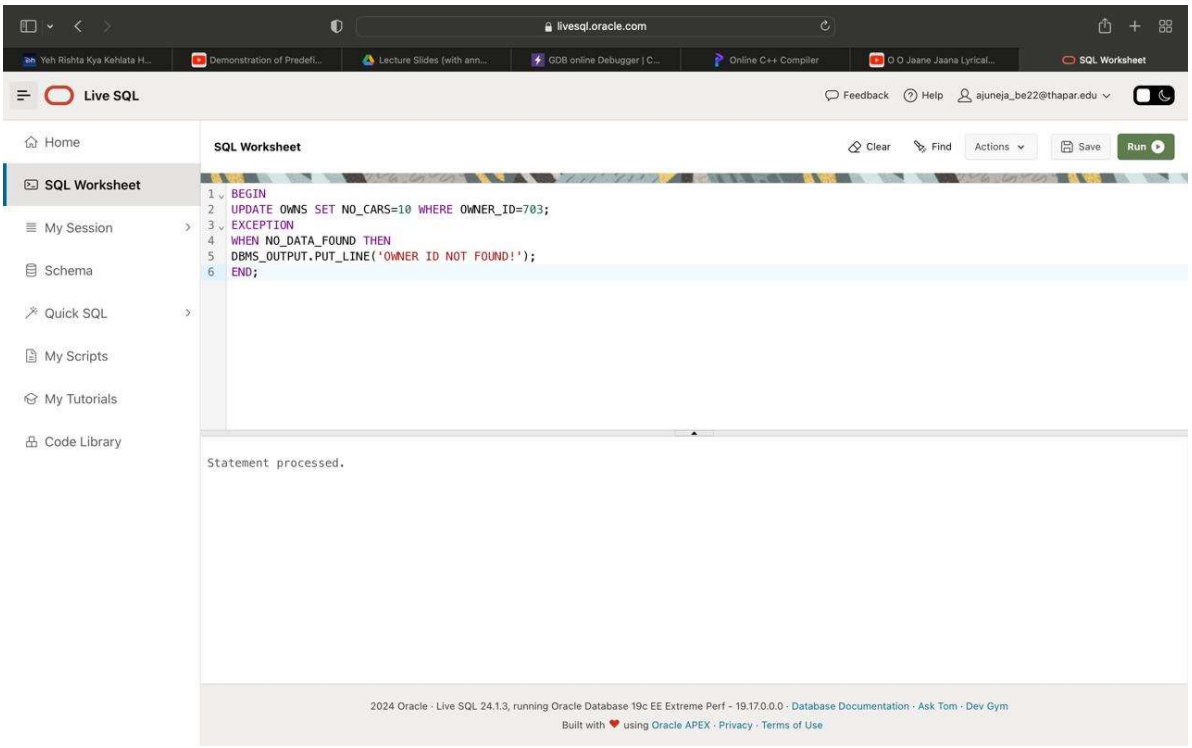
EXCEPTION HANDLING WHEN UNIQUENESS PROPERTY OF PRIMARY KEY IS VIOLATED

```
BEGIN
INSERT INTO CUSTOMER_SERVICE VALUES(501,'ASMI','JUNEJA');
EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
DBMS_OUTPUT.PUT_LINE('INSERT STATEMENT IS VIOLATING UNIQUE
CONSTRAINT!');
END;
```



EXCEPTION HANDLING WHEN NO DATA IS FOUND

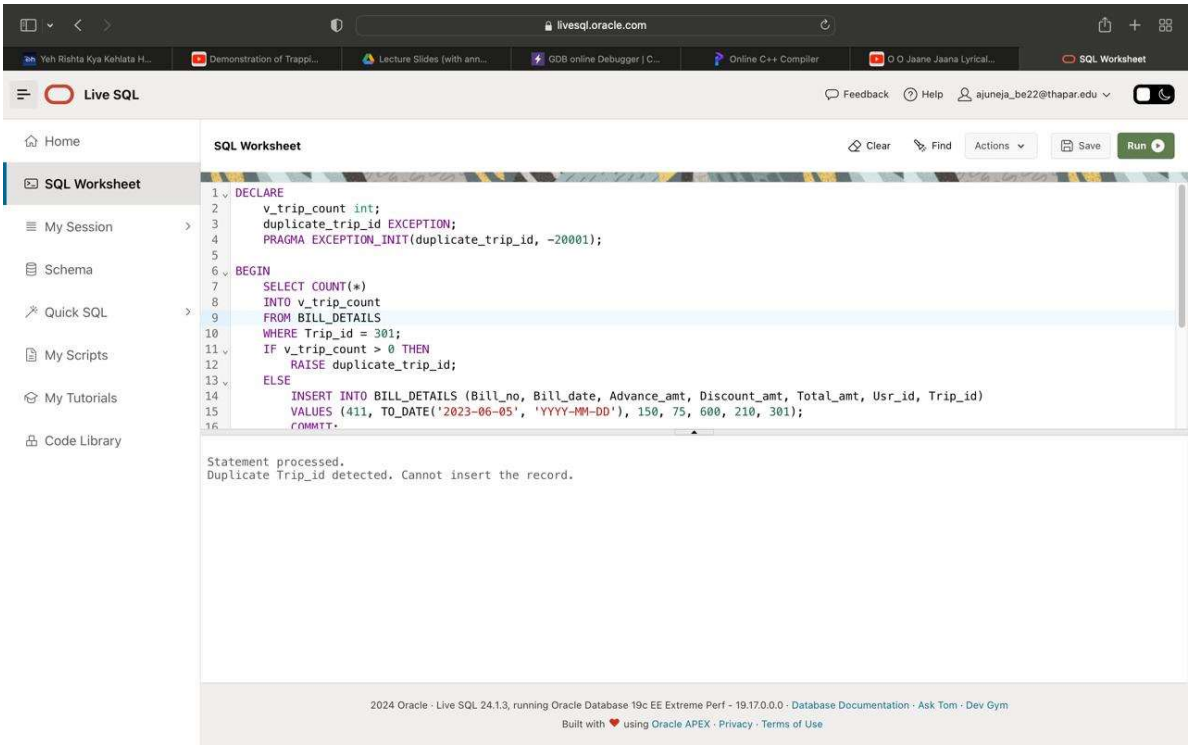
```
BEGIN
UPDATE OWNS SET NO_CARS=10 WHERE OWNER_ID=703;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('OWNER ID NOT FOUND!');
END;
```



EXCEPTION HANDLING TO HANDLE DUPLICATE TRIP_ID

```
DECLARE
   v_trip_count int;
   duplicate_trip_id EXCEPTION;
   PRAGMA EXCEPTION_INIT(duplicate_trip_id, -20001);

BEGIN
   SELECT COUNT(*)
   INTO v_trip_count
   FROM BILL_DETAILS
   WHERE Trip_id = 301;
   IF v_trip_count > 0 THEN
      RAISE duplicate_trip_id;
   ELSE
      INSERT INTO BILL_DETAILS (Bill_no, Bill_date, Advance_amt, Discount_amt,
Total_amt, Usr_id, Trip_id)
      VALUES (411, TO_DATE('2023-06-05', 'YYYY-MM-DD'), 150, 75, 600, 210, 301);
      COMMIT;
   END IF;
EXCEPTION
   WHEN duplicate_trip_id THEN
      DBMS_OUTPUT.PUT_LINE('Duplicate Trip_id detected. Cannot insert the
record.');
```

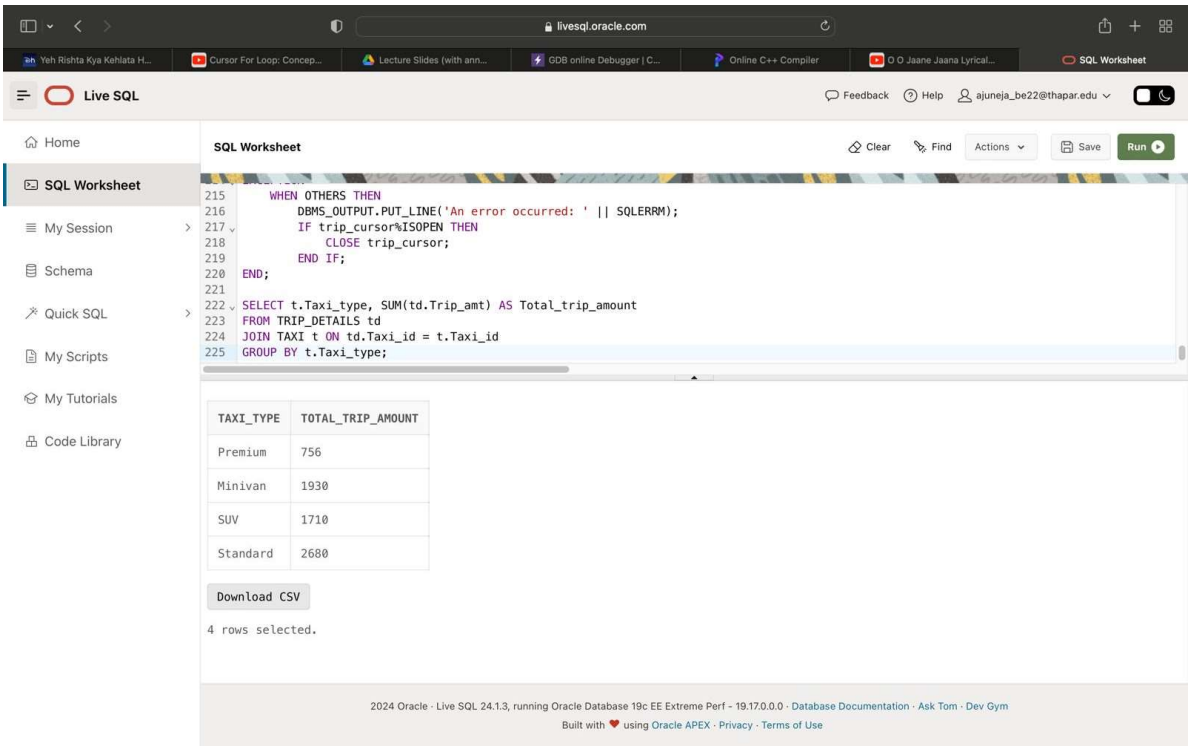
```
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END;
```



QUERIES

1. DISPLAY THE TOTAL TRIP AMOUNTS FOR EACH TAXI TYPE.

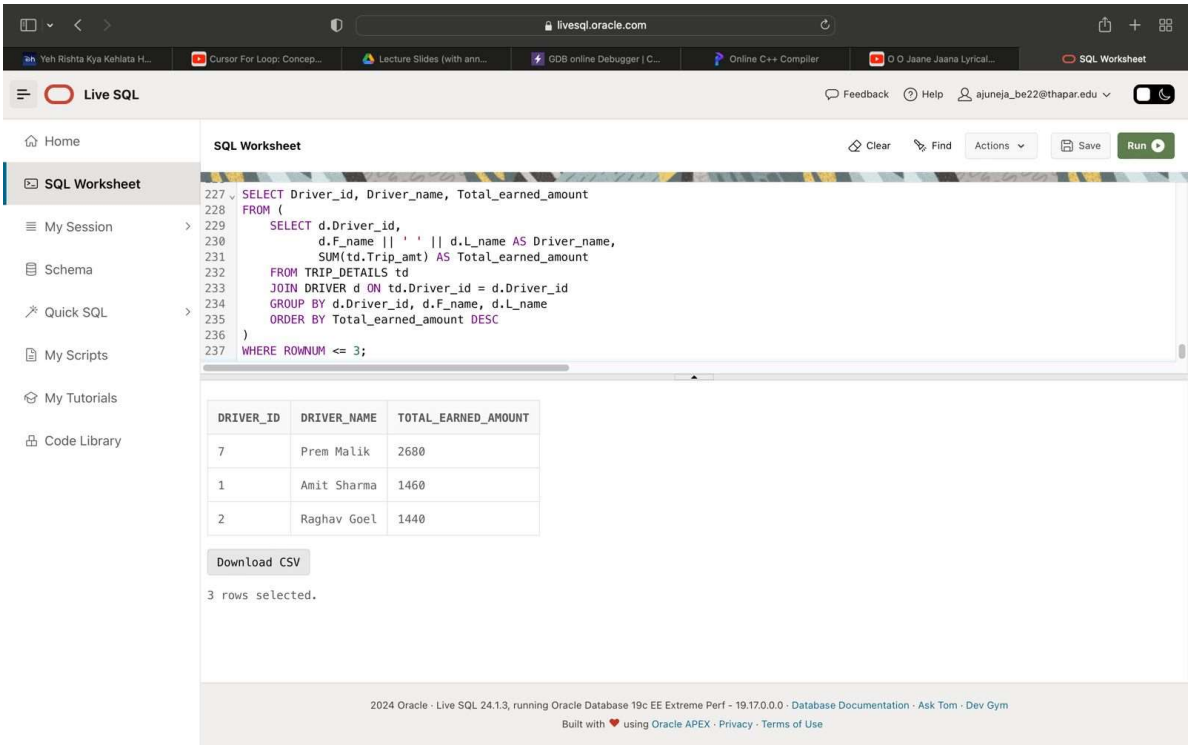
```
SELECT t.Taxi_type, SUM(td.Trip_amt) AS Total_trip_amount
FROM TRIP_DETAILS td
JOIN TAXI t ON td.Taxi_id = t.Taxi_id
GROUP BY t.Taxi_type;
```



2. DISPLAY THE TOP 3 DRIVERS BASED ON THE TOTAL TRIP AMOUNTS THEY EARNED

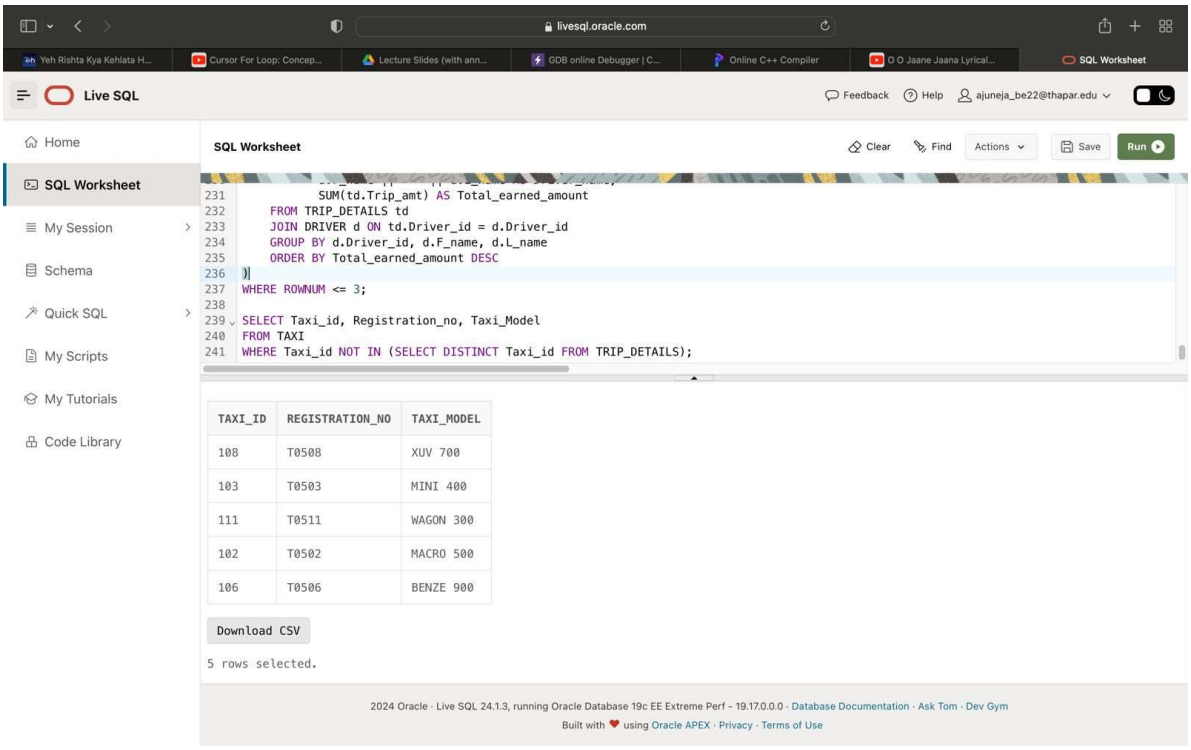
```
SELECT Driver_id, Driver_name, Total_earned_amount
FROM (
  SELECT d.Driver_id,
      d.F_name || ' ' || d.L_name AS Driver_name,
      SUM(td.Trip_amt) AS Total_earned_amount
  FROM TRIP_DETAILS td
```

```
    JOIN DRIVER d ON td.Driver_id = d.Driver_id
    GROUP BY d.Driver_id, d.F_name, d.L_name
    ORDER BY Total_earned_amount DESC
)
WHERE ROWNUM <= 3;
```



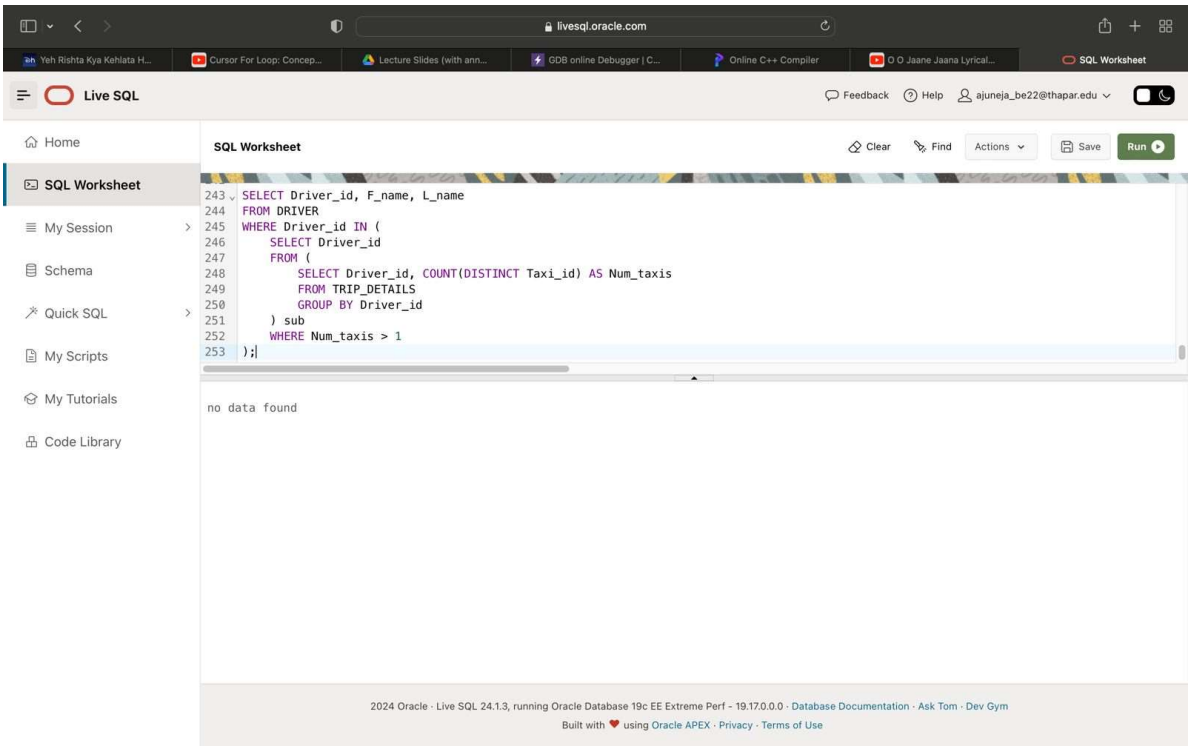
3. DISPLAY THE TAXIS THAT HAVE NOT BEEN USED FOR ANY TRIPS

```
SELECT Taxi_id, Registration_no, Taxi_Model
FROM TAXI
WHERE Taxi_id NOT IN (SELECT DISTINCT Taxi_id FROM TRIP_DETAILS);
```



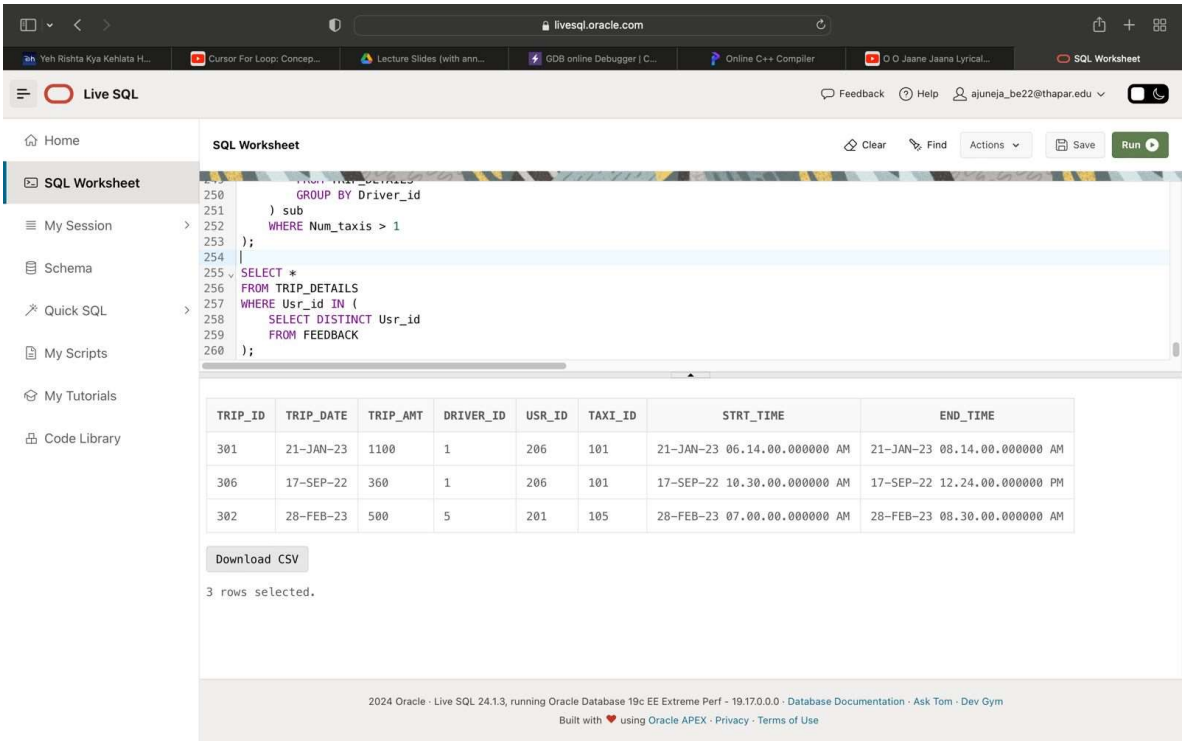
4. DISPLAY THE DRIVERS WHO HAVE COMPLETED TRIPS ON MORE THAN ONE TAXI

```
SELECT Driver_id, F_name, L_name
FROM DRIVER
WHERE Driver_id IN (
   SELECT Driver_id
   FROM (
      SELECT Driver_id, COUNT(DISTINCT Taxi_id) AS Num_taxis
      FROM TRIP_DETAILS
      GROUP BY Driver_id
   ) sub
   WHERE Num_taxis > 1
);
```



5. LIST THE TRIP DETAILS FOR USERS WHO HAVE PROVIDED FEEDBACK

```
SELECT *
FROM TRIP_DETAILS
WHERE Usr_id IN (
   SELECT DISTINCT Usr_id
   FROM FEEDBACK
);
```


livesql.oracle.com
Live SQL
Feedback
Help
ajuneja_be22@thapar.edu
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
GROUP BY Driver_id
) sub
WHERE Num_taxis > 1
);
SELECT *
FROM TRIP_DETAILS
WHERE Usr_id IN (
SELECT DISTINCT Usr_id
FROM FEEDBACK
);
TRIP_ID TRIP_DATE TRIP_AMT DRIVER_ID USR_ID TAXI_ID STRT_TIME END_TIME
301 21-JAN-23 1100 1 206 101 21-JAN-23 06.14.00.000000 AM 21-JAN-23 08.14.00.000000 AM
306 17-SEP-22 360 1 206 101 17-SEP-22 10.30.00.000000 AM 17-SEP-22 12.24.00.000000 PM
302 28-FEB-23 500 5 201 105 28-FEB-23 07.00.00.000000 AM 28-FEB-23 08.30.00.000000 AM
Download CSV
3 rows selected.
2024 Oracle · Live SQL 24.1.3, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with using Oracle APEX · Privacy · Terms of Use

CONCLUSION

In conclusion, a well-structured taxi database management system serves as the backbone of efficient taxi services, providing essential functionalities such as booking management, driver allocation, billing, and feedback handling. By centralizing and organizing data pertaining to trips, drivers, customers, and financial transactions, the system enables streamlined operations and effective decision-making. Through features like trip history analysis and customer feedback aggregation, taxi companies can continuously refine their services, enhancing customer satisfaction and loyalty. Moreover, the system supports scalability, allowing taxi businesses to adapt to changing demands and expand their operations seamlessly. Overall, a robust taxi database management system is pivotal in optimizing service quality, maximizing operational efficiency, and sustaining competitiveness in the ever-evolving transportation industry.

REFERENCES

1. Database Management Systems (Book by Raghu Ramakrishnan & Johannes Gehrke)
2. https://youtube.com/playlist?list=PLVCEF4zOWjkifoEBe_KFovoMXe-QfsPai&si=rI-cV4yOoOiv5euW (PL/SQL by Prateek Bhatia)
3. https://www.geeksforgeeks.org/dbms/
4. https://nesoacademy.org/cs/08-database-management-system
5. http://draw.io/