SQL ASSIGNMENT

Submitted By:

G.S.Sameer Basha

Project Trainee, KPI PARTNERS

Email: [gadivemula.basha@kpipartners.com](mailto:gadivemula.basha@kpipartners.com)

**NEW YORK CITY TAXI AND LIMOUSINE COMMISSION**

**TABLE OF CONTENT**

**CHAPTER. 1. INTRODUCTION**

**CHAPETR. 2. A BRIEF DESCRIPTION OF THE DATA USED**

**CHAPETR. 3. QUERIES AND RESULTS**

**CHAPTER 1**

**INTRODUCTION**

The New York City Taxi and Limousine Commission (TLC), created in 1971, is the agency responsible for licensing and regulating New York City's Medallion (green) taxi cabs, for-hire vehicles (community-based liveries, black cars and luxury limousines), commuter vans, and paratransit vehicles. Over 200,000 TLC licensees complete approximately 1,000,000 trips each day.

To operate for hire, drivers must first undergo a background check, have a safe driving record, and complete 24 hours of driver training. TLC-licensed vehicles are inspected for safety and emissions at TLC's Woodside Inspection Facility. It is a priority of the New York City Taxi and Limousine Commission to provide safe, reliable transportation options for all New Yorkers and to recognize and address the needs of our licensees. TLC supports and contributes to city-wide efforts of traffic safety, accessibility and technological improvements. Policy researchers at the TLC use data generated by licenses to observe changing trends in the industry and inform decisions made by agency and the City.

**CHAPTER 2**

**A BRIEF DESCRIPTION OF THE DATA USED**

**VendorID:**

A code indicating the LPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.

**lpep\_pickup\_datetime:**

The date and time when the meter was engaged.

**lpep\_dropoff\_datetime:**

The date and time when the meter was disengaged.

**Passenger\_count:**

The number of passengers in the vehicle. This is a driver-entered value.

**Trip\_distance:**

The elapsed trip distance in miles reported by the taximeter.

**PULocationID:**

TLC Taxi Zone in which the taximeter was engaged.

**DOLocationID:**

TLC Taxi Zone in which the taximeter was disengaged.

**RateCodeID:**

The final rate code in effect at the end of the trip. 1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride.

**Store\_and\_fwd\_flag:**

This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server. Y= store and forward trip, N= not a store and forward trip

**Payment\_type:**

A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip.

**Fare\_amount:**

The time-and-distance fare calculated by the meter. Extra Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges. MTA\_tax $0.50 MTA tax that is automatically triggered based on the metered rate in use. Improvement\_surcharge $0.30 improvement surcharge assessed on hailed trips at the flag drop. The improvement surcharge began being levied in 2015.

**Tip\_amount:**

This field is automatically populated for credit card tips. Cash tips are not included.

**Tolls\_amount:**

Total amount of all tolls paid in trip.

**Total\_amount:**

The total amount charged to passengers. Does not include cash tips.

**Trip\_type:**

A code indicating whether the trip was a street-hail or a dispatch that is automatically assigned based on the metered rate in use but can be altered by the driver. 1= Street-hail 2= Dispatch

In this analysis for storage and querying the green taxi trip data accumulated for the year 2019, we have used the following

**CHAPETR. 3**

**QUERIES AND RESULTS**

**Create the table structure with appropriate data types before loading data**

create table green\_tr

(

"VendorID" float,

lpep\_pickup\_datetime varchar2(40),

lpep\_dropoff\_datetime varchar2(40),

store\_and\_fwd\_flag varchar2(1) default 'N',

"RatecodeID" float,

"PULocationID" float,

"DOLocationID" float,

passenger\_count float,

trip\_distance float,

fare\_amount float,

extra float,

mta\_tax float,

tip\_amount float,

tolls\_amount float,

ehail\_fee float ,

improvement\_surcharge float,

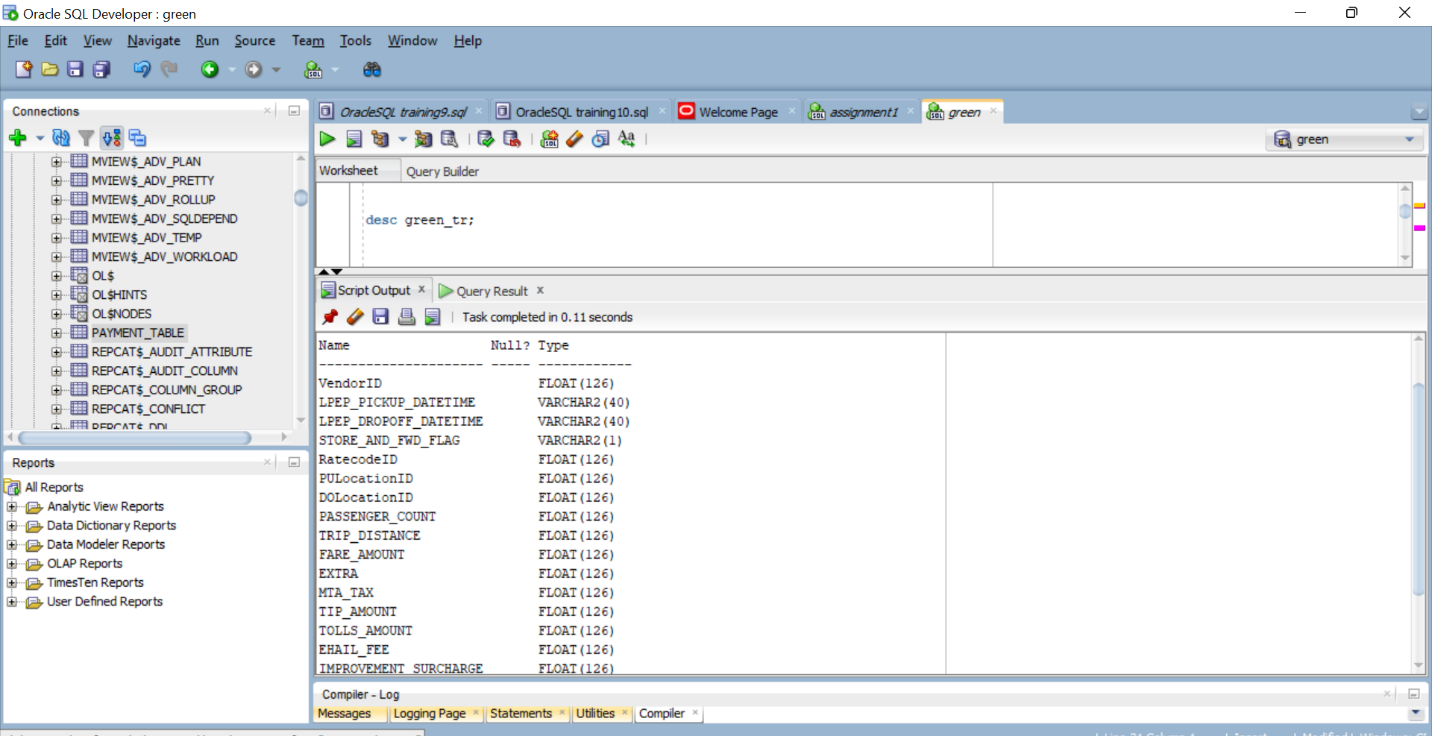
total\_amount float,

payment\_type float,

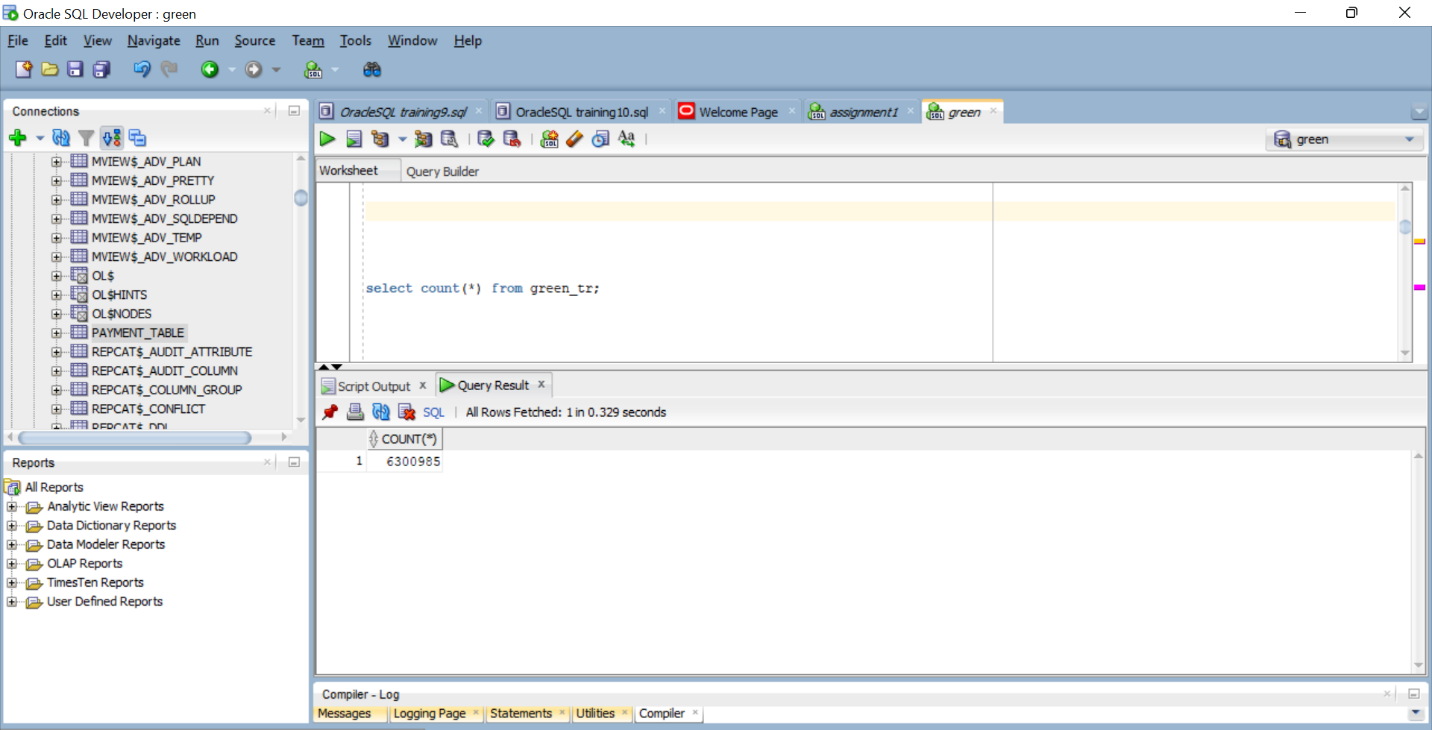
trip\_type float,

congestion\_surcharge float

);



select count(\*) from green\_tr;



CREATE TABLE taxi\_lookup1

(

"LocationID" NUMBER primary key,

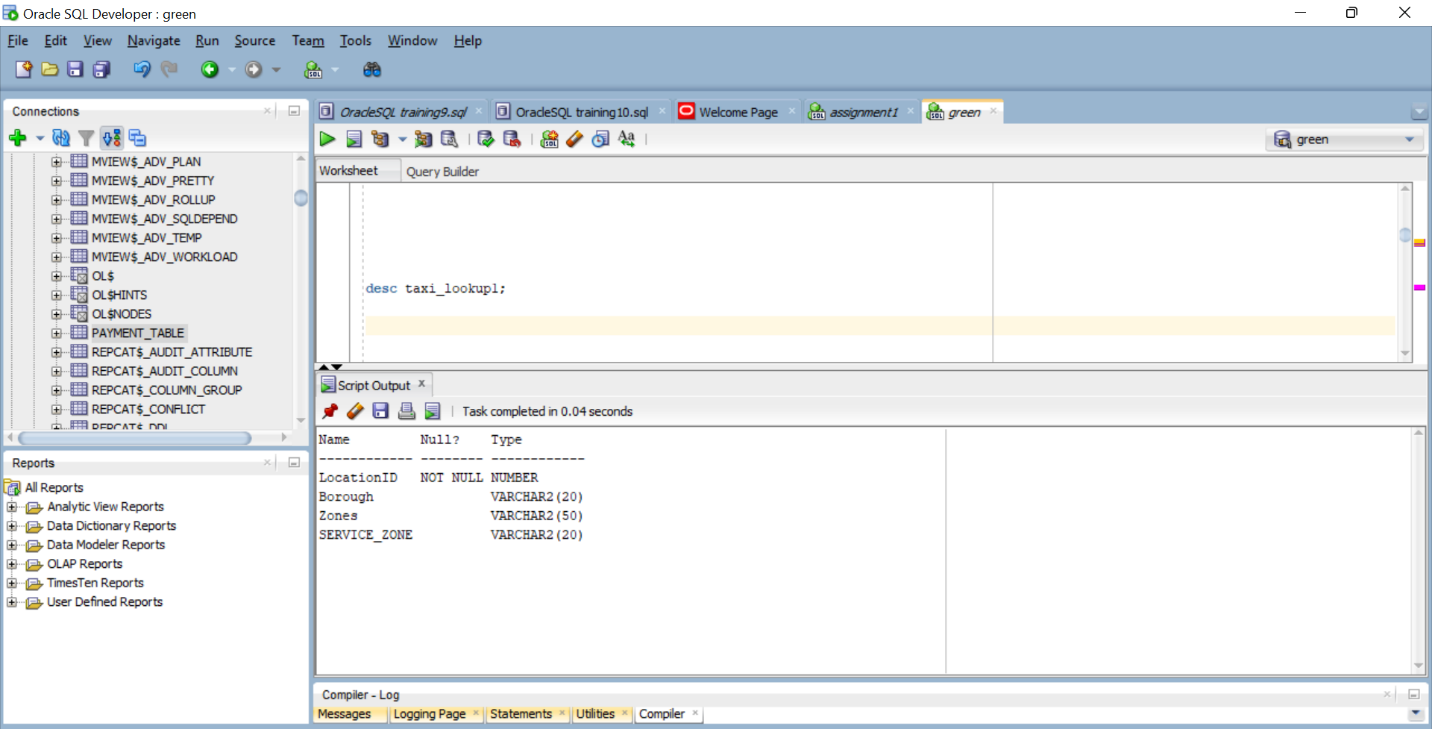
"Borough" VARCHAR2(20),

"Zones" VARCHAR2(50),

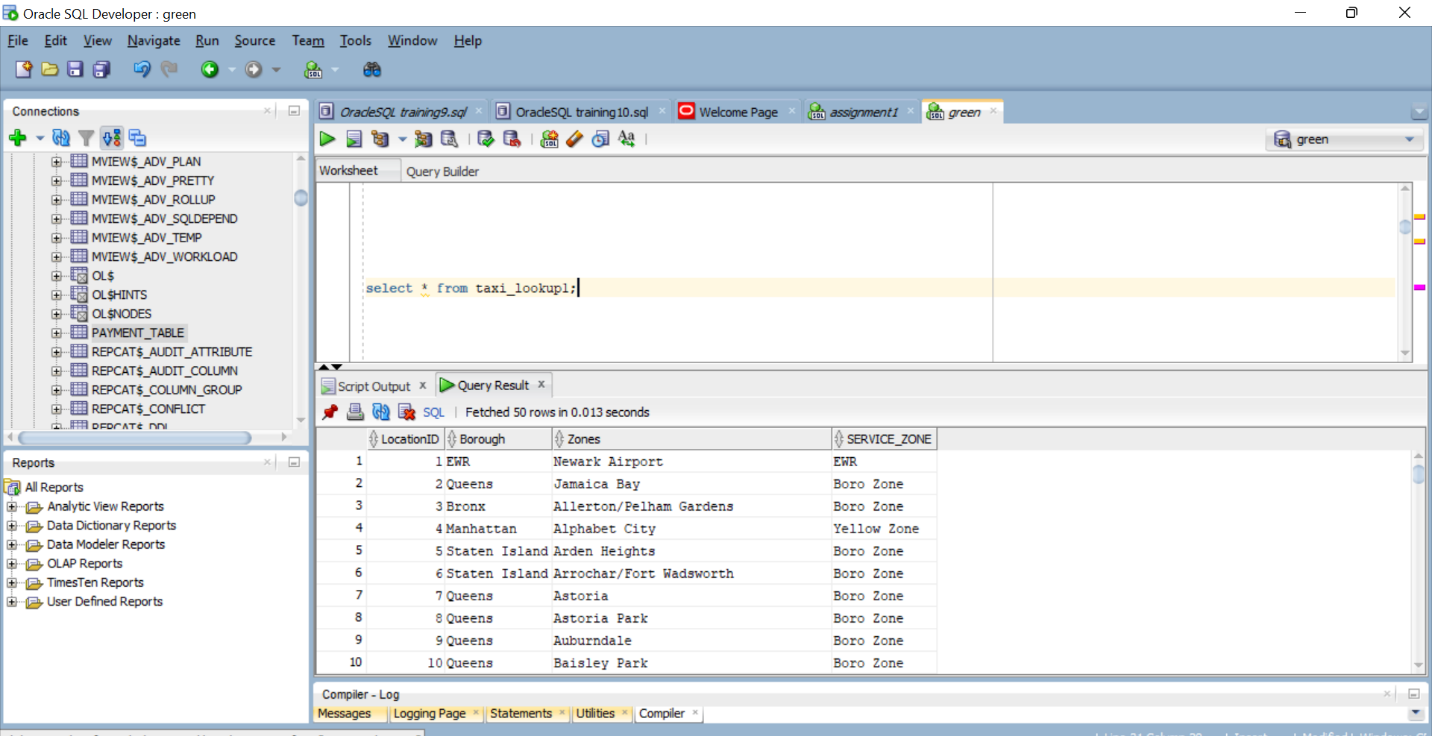
service\_zone VARCHAR2(20)

);

Desc taxi\_lookup1;

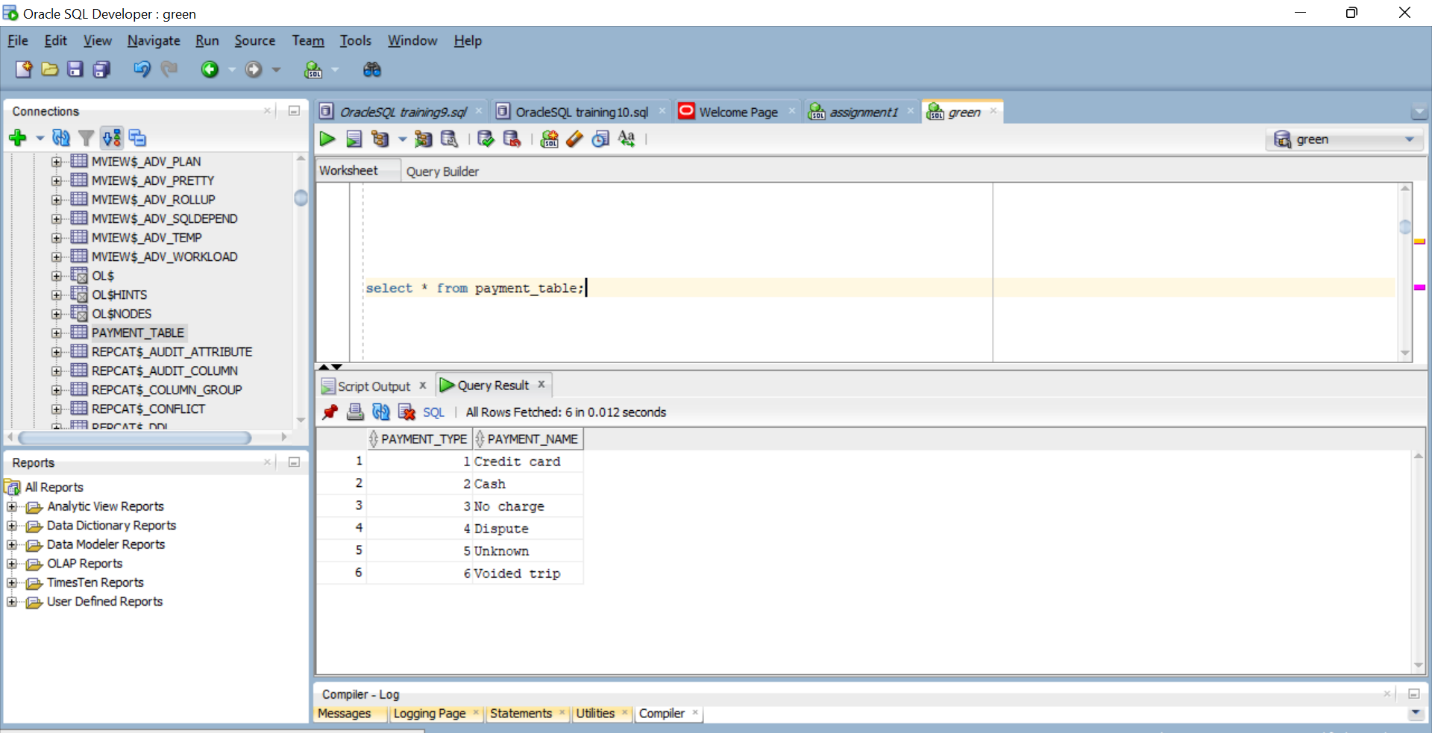


select \* from taxi\_lookup1;



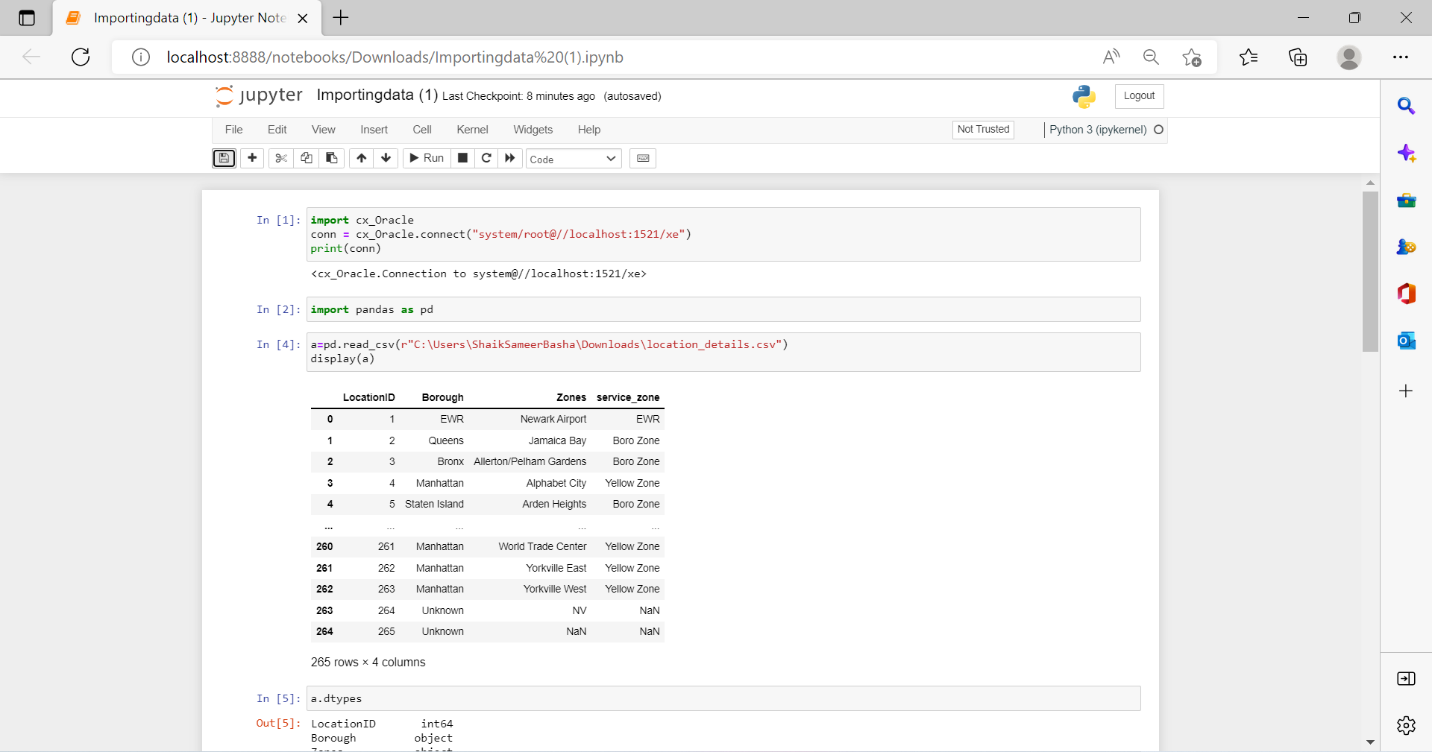
create table payment\_table(payment\_type number(2), payment\_name varchar2(30));

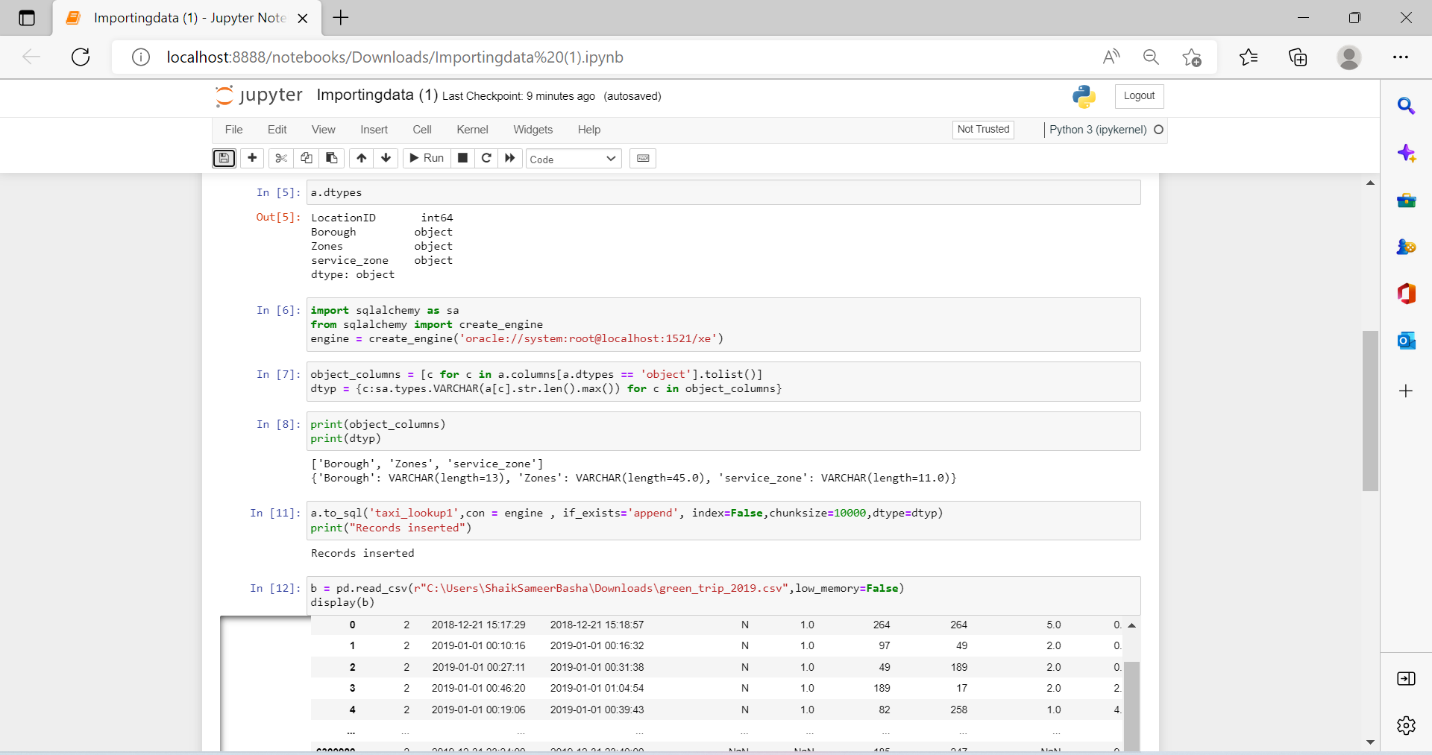
select \* from payment\_table;

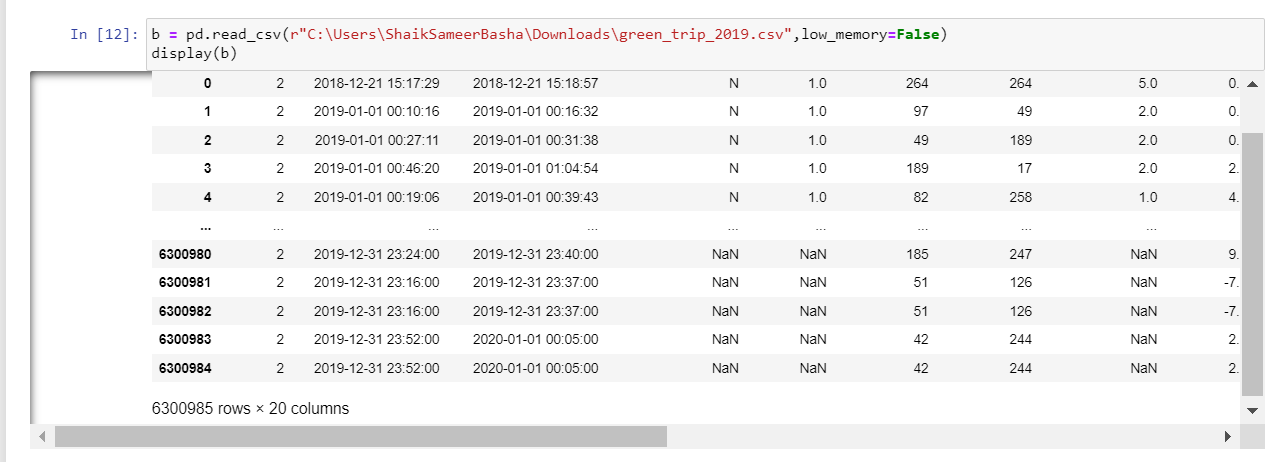


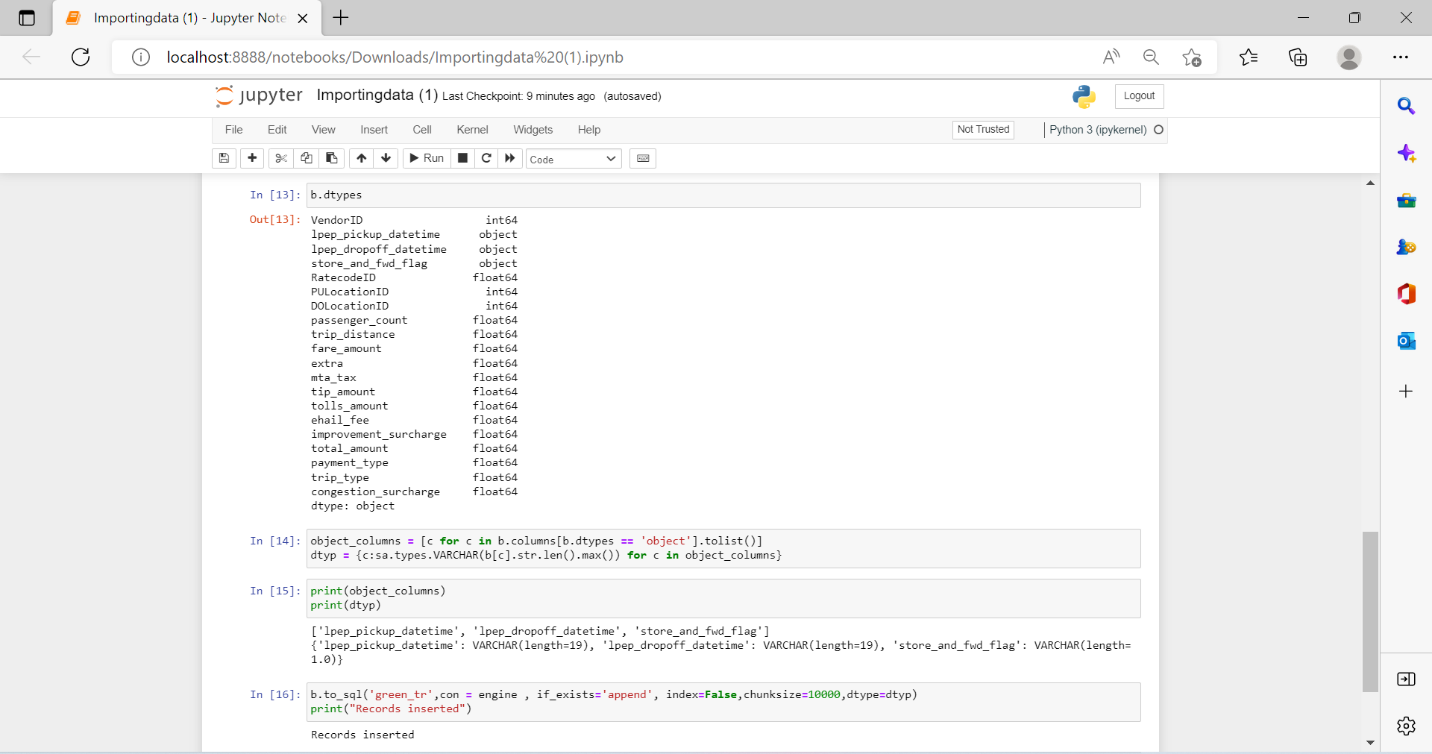
**Reading the csv file using pandas and storing in a variable and exporting the read data to oracle database using sqlalchemy module.**

1. Use python to load the all the 2019 files to an Oracle Database table









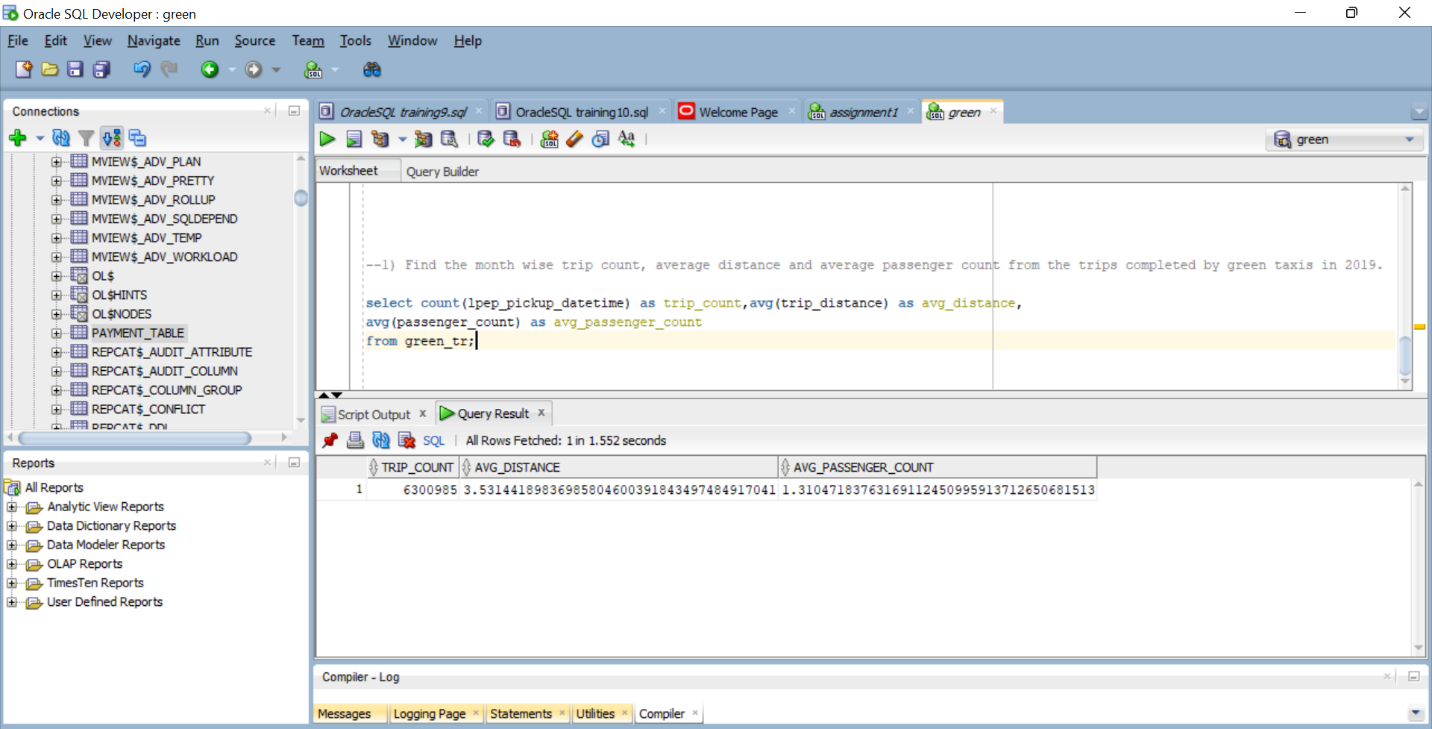
**QUERIES**

A) Find the month wise trip count, average distance and average passenger count from the trips completed by green taxis in 2019.

select count(lpep\_pickup\_datetime) as trip\_count,avg(trip\_distance) as avg\_distance,

avg(passenger\_count) as avg\_passenger\_count

from green\_tr;



B) Find out the five busiest routes served by the green taxis during 2019. The name of start and drop points to be provided.

select \* from

(

with TripT as

(

select TL1."Zones" || ',' || TL1."Borough" as "Pick Point", TL2."Zones" || ',' || TL2."Borough" as "Drop Point"

from green\_tr

inner join taxi\_lookup1 TL1 on TL1."LocationID" =green\_tr."PULocationID"

inner join taxi\_lookup1 TL2 on TL2."LocationID"=green\_tr."DOLocationID"

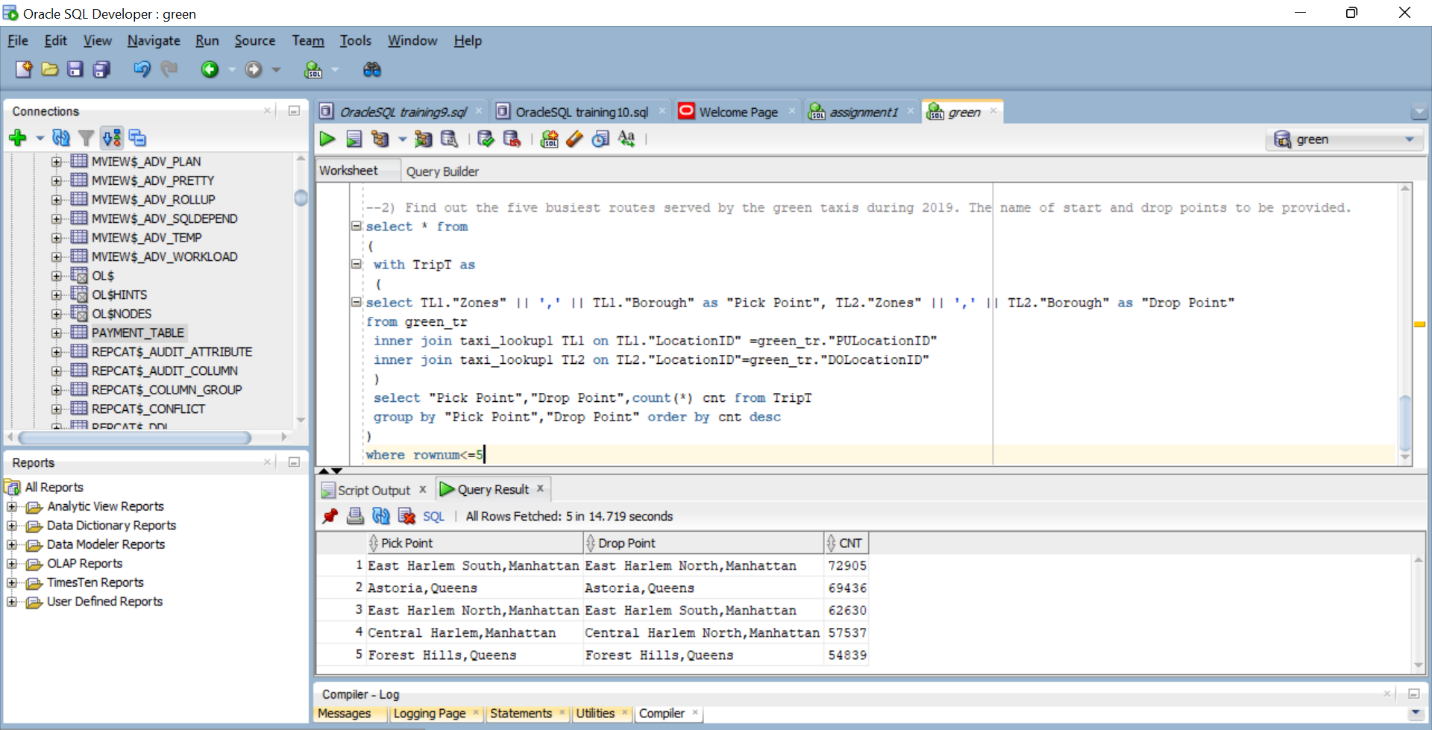
)

select "Pick Point","Drop Point",count(\*) cnt from TripT

group by "Pick Point","Drop Point" order by cnt desc

)

where rownum<=5



C) What are the top 3 busiest hours of the day for the taxis?

SELECT Hours AS busiest\_hour, pickup\_count

FROM

(SELECT

EXTRACT(HOUR FROM TO\_TIMESTAMP(lpep\_pickup\_datetime, 'YYYY-MM-DD HH24:MI:SS')) AS Hours,

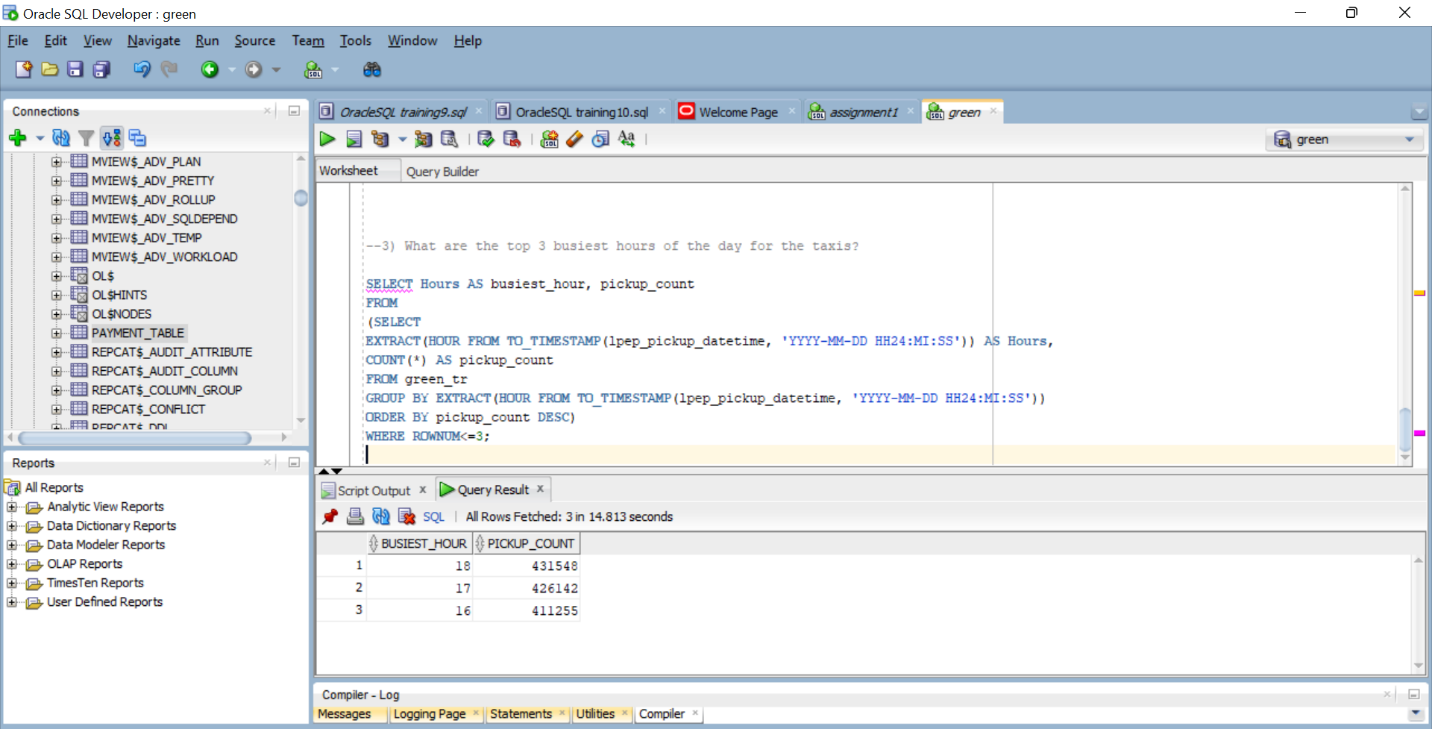
COUNT(\*) AS pickup\_count

FROM green\_tr

GROUP BY EXTRACT(HOUR FROM TO\_TIMESTAMP(lpep\_pickup\_datetime, 'YYYY-MM-DD HH24:MI:SS'))

ORDER BY pickup\_count DESC)

WHERE ROWNUM<=3;



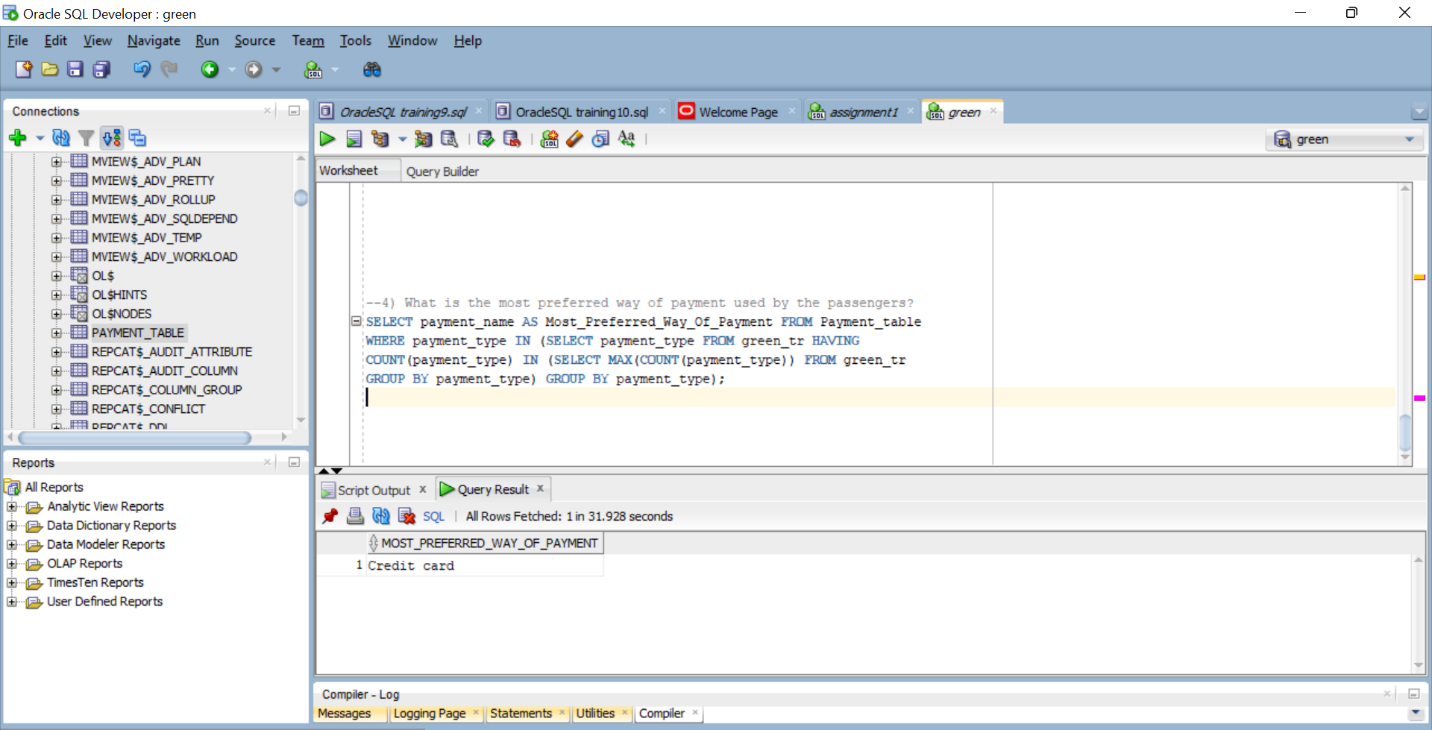
D) What is the most preferred way of payment used by the passengers?

SELECT payment\_name AS Most\_Preferred\_Way\_Of\_Payment FROM Payment\_table

WHERE payment\_type IN (SELECT payment\_type FROM green\_tr HAVING

COUNT(payment\_type) IN (SELECT MAX(COUNT(payment\_type)) FROM green\_tr

GROUP BY payment\_type) GROUP BY payment\_type);



E) Write a PL/SQL block to read through each record and update ehail\_fee to 0.5 (capture the time taken for execution)

create or replace procedure Update\_ehail\_fee

is

begin

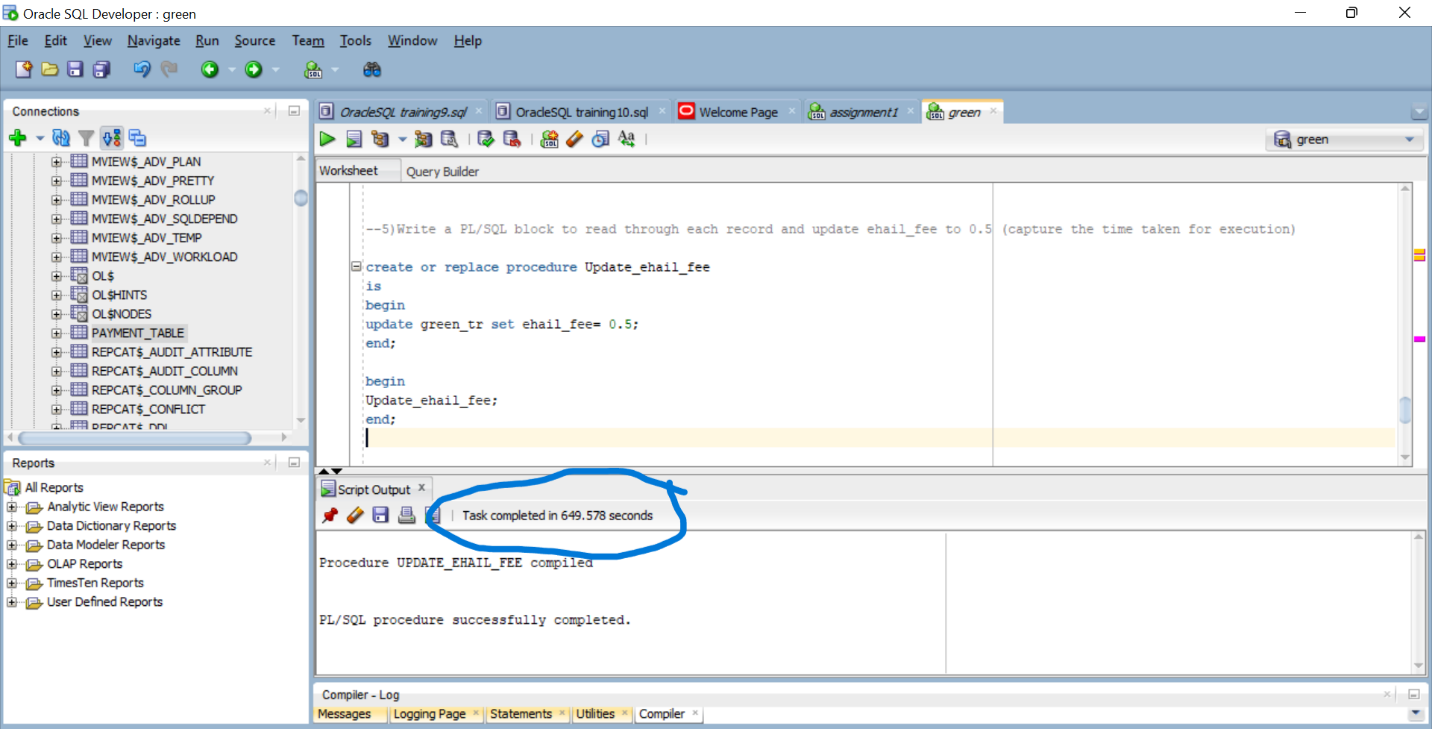
update green\_tr set ehail\_fee= 0.5;

end;

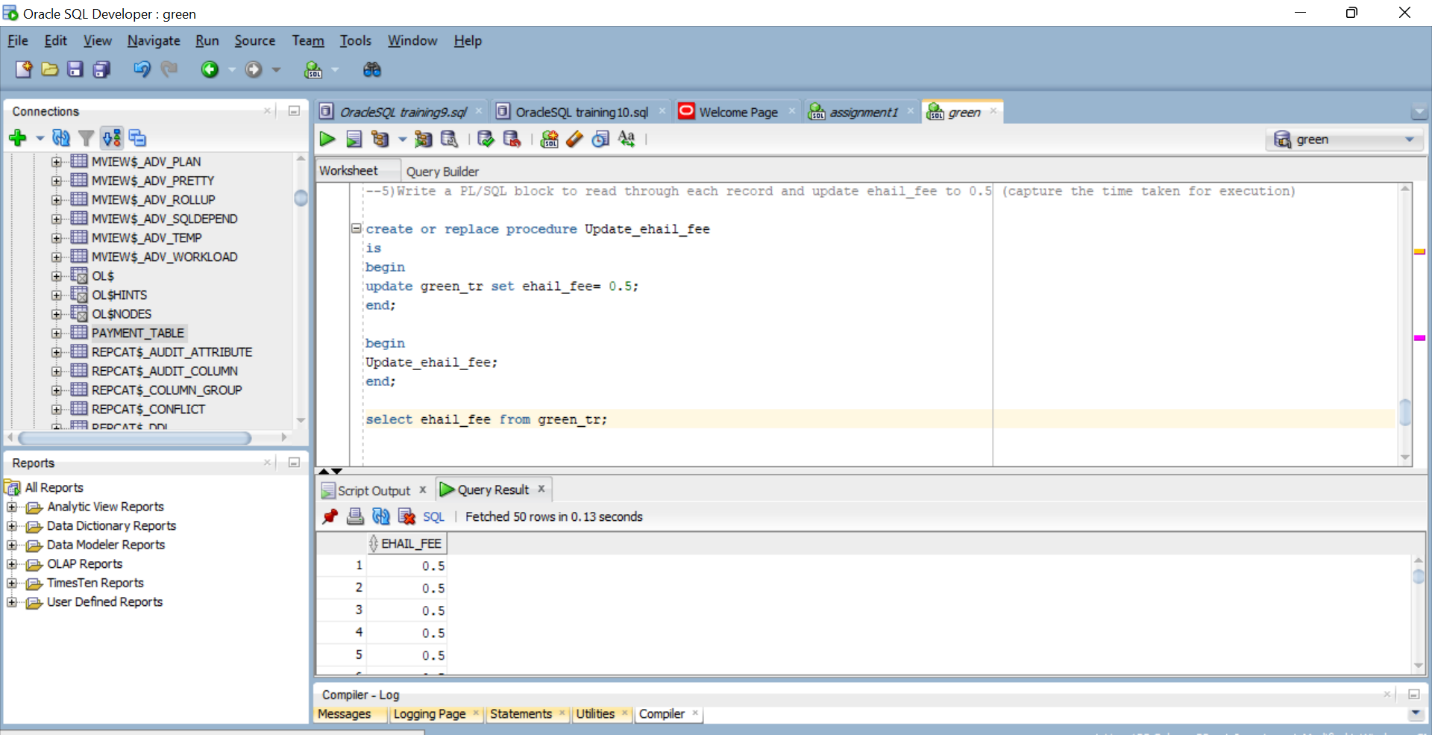
begin

Update\_ehail\_fee;

end;

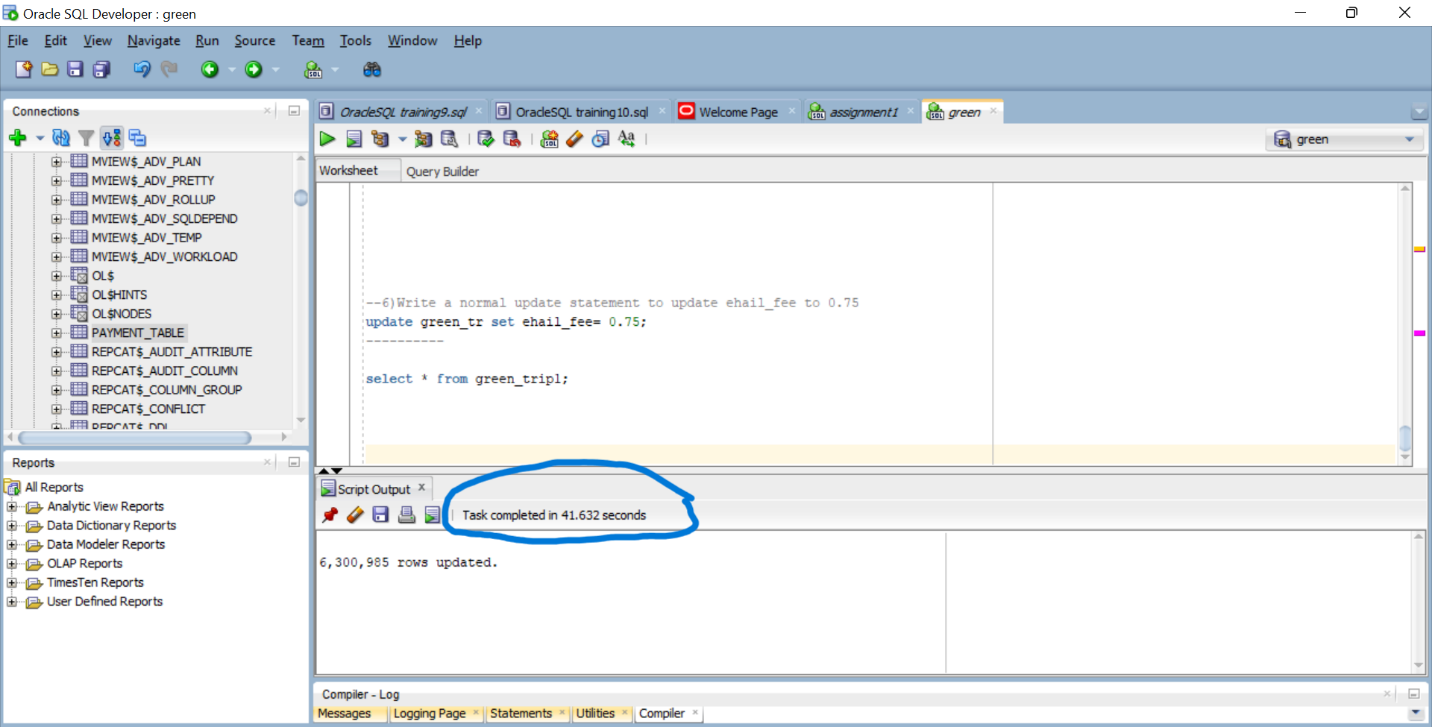


select ehail\_fee from green\_tr;

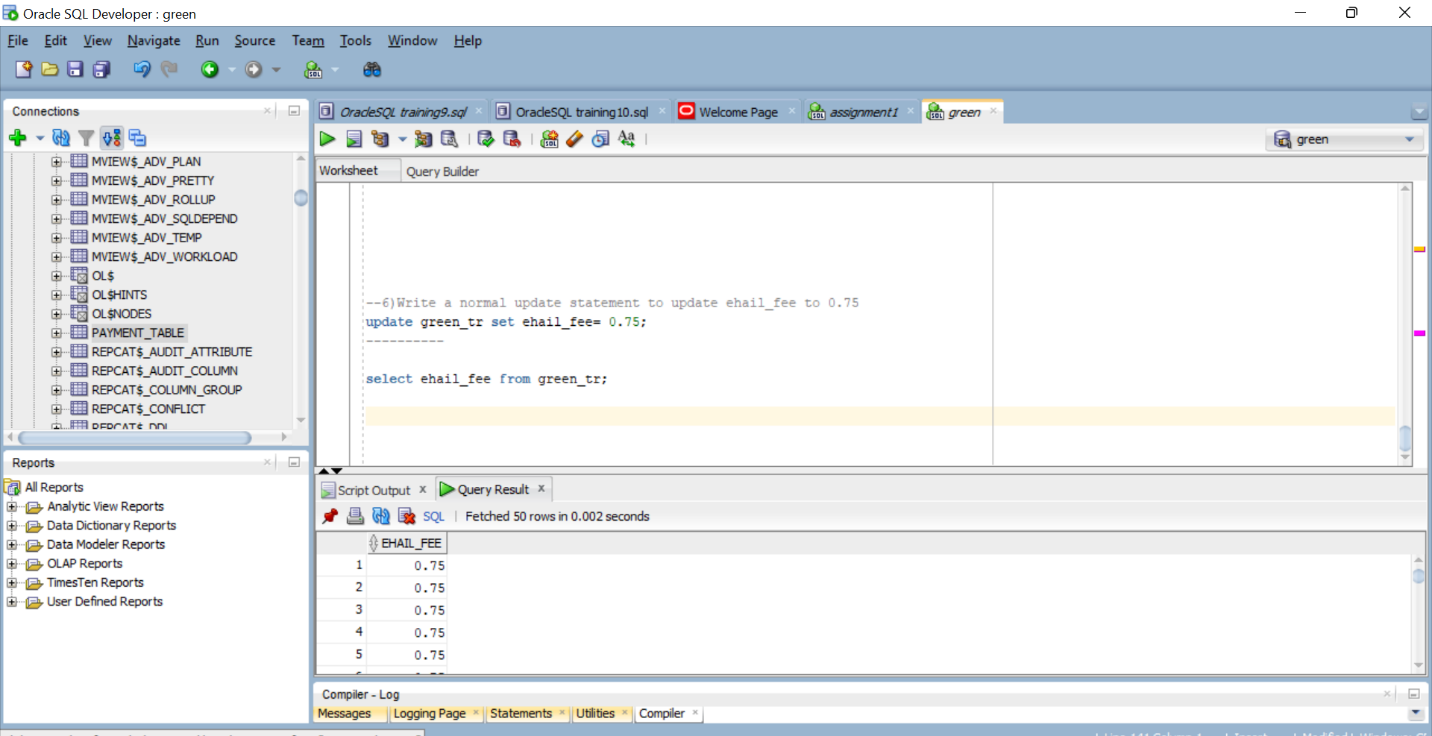


F) Write a normal update statement to update ehail\_fee to 0.75

update green\_tr set ehail\_fee= 0.75;



select ehail\_fee from green\_tr;



1. Identify the time taken by e and f and provide your analysis on why each step took more/less time compared to other

Both E & F updated 63,00,985 records separately. Time taken by E is 649.578 Seconds

Where F took 41.632 seconds.

When we compare the time taken by E & F. E is Too slow & F is faster.

E consists the procedure which stored in the sql server takes much time to execute.

While F with normal sql update statement took less time