| Lab – 1.1 | Section1\_Priyanka  Railway\_Reservation\_Database System |
| --- | --- |
| IT214 Database Management System, Autumn’2022; Instructor: minal\_bhise@daiict | |

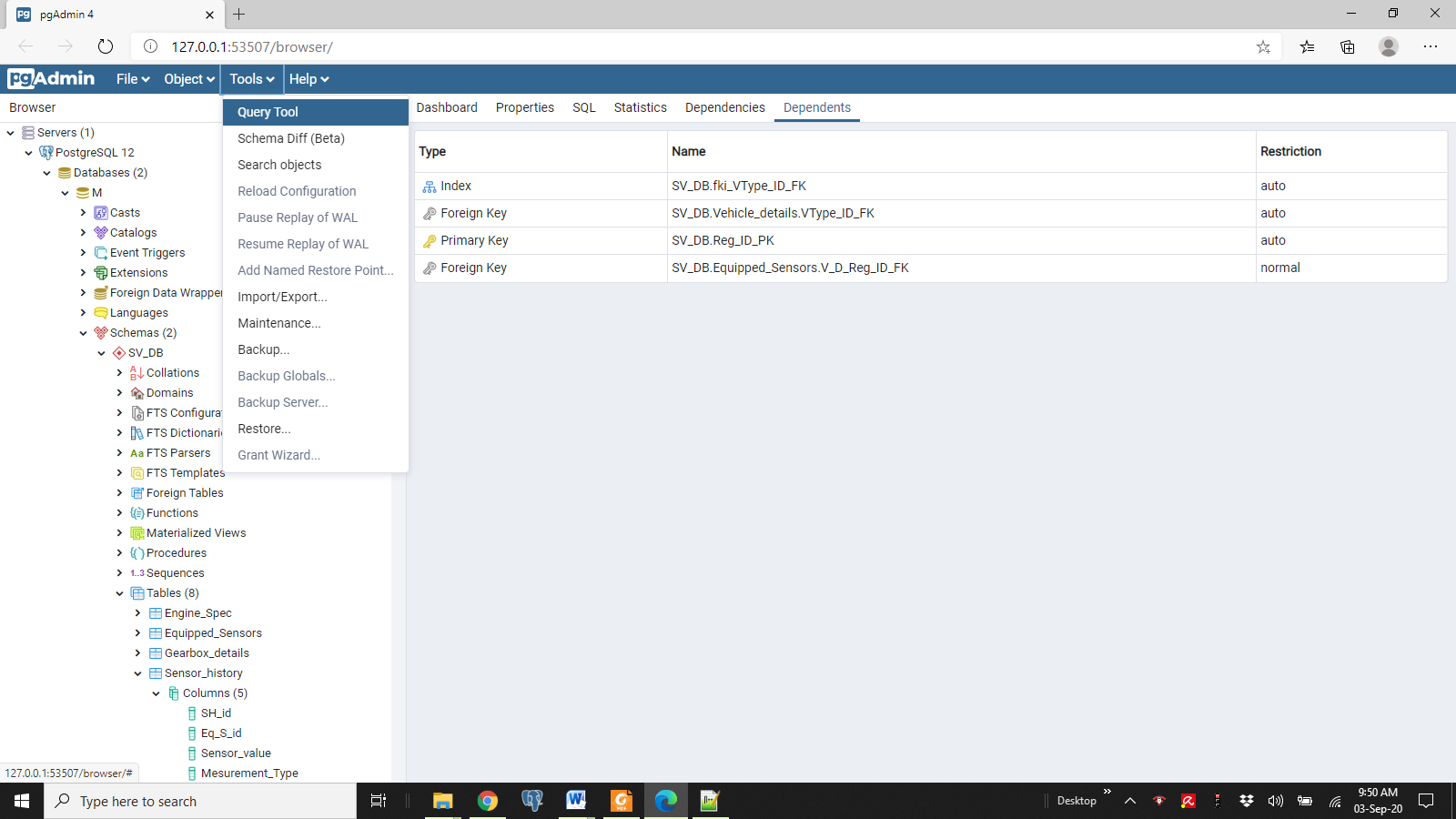
**Objectives**: I) Create tables using below given schema & DDL Scripts in Appendix A.

II) Load data into tables using csv files given in your classroom.

III) Run simple queries.

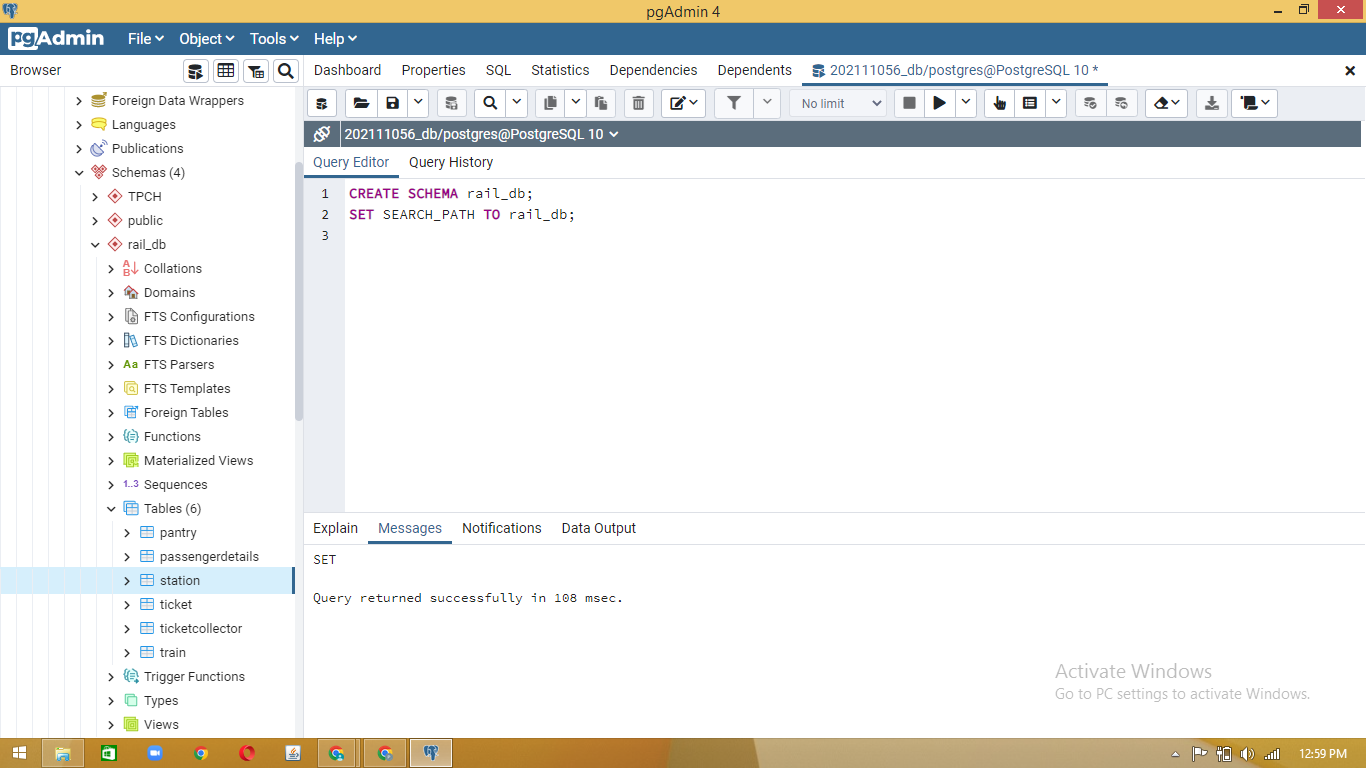
**Submission**: Each student needs to Upload a **single .pdf** file which will contain following things for all the queries listed in (III).

1. English query and SQL Query in the given sequence.
2. Screenshot of results.
3. Count of tuples in the results.
4. **Create tables using below given schema:** 
   1. Expand your Database => Open Query Tool window from Tool. or From SQL Shell.

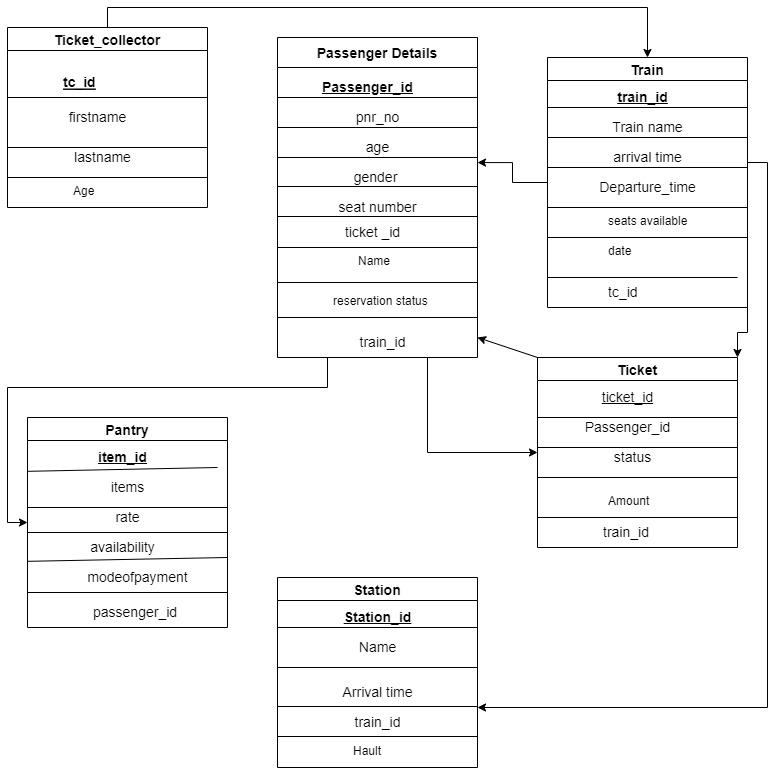


* 1. Type following to create Schema. –

**CREATE SCHEMA rail\_db;  
SET SEARCH\_PATH TO rail\_db;**



* 1. Table Schema Diagram. Find the DDL Scripts after the queries section III.

****

1. **Load data into tables using csv files given on classroom file lab1.zip**

COPY "passengerdetails" FROM 'G:\TAdbms\S8-G3,G6-Priyanka - passenger\_details.csv' CSV HEADER;

1. **Run simple queries.**

Query 1: Show all the details of the ticket collector.

Query 2: List the ticket id whose price is greater than Rs3000.

Query 3: Show all the food items that are not available.

Query 4: Count the number of female passengers.

Query 5: Find out the smallest age of passenger travelling.

Query 6: Find out the eldest tc among all the ticket collectors.

Query 7: Write a query to print all the food items being served on the train.

Query 8: Print the name of trains where numb of seats available is greater than 5

Query 9: Print the number of rows present in the passenger table

Query 10: Find the ticket id of the costliest ticket.

Query 11: Show the second highest priced ticket\_id

Query 12: Print the name of passengers whose ticket is confirmed

Query 13: Print all the names of stations.

Query 14: Print the name of all the ticket collectors who are below the age of 40.

Query 15: Write a query to print the number of foods that were available & their mode of payment was online.

1. **Appendix A: DDL Scripts to create tables.**

-----------------------------------------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS rail\_db.pantry

(

item\_id integer NOT NULL,

item character(30) COLLATE pg\_catalog."default",

rate integer,

availability character(30) COLLATE pg\_catalog."default",

payment\_mode character(30) COLLATE pg\_catalog."default",

passenger\_id integer,

CONSTRAINT pantry\_pkey PRIMARY KEY (item\_id),

CONSTRAINT passenger\_id FOREIGN KEY (passenger\_id)

REFERENCES rail\_db.passengerdetails (passenger\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS rail\_db.pantry

OWNER to postgres;

CREATE TABLE IF NOT EXISTS rail\_db.passengerdetails

(

passenger\_id integer NOT NULL,

pnr\_no integer,

age integer,

gender character(30) COLLATE pg\_catalog."default",

seat\_no integer,

ticket\_id integer,

name character(50) COLLATE pg\_catalog."default",

reservation\_status character(40) COLLATE pg\_catalog."default",

train\_id integer,

CONSTRAINT passengerdetails\_pkey PRIMARY KEY (passenger\_id),

CONSTRAINT ticket\_id FOREIGN KEY (ticket\_id)

REFERENCES rail\_db.ticket (ticket\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT train\_id FOREIGN KEY (train\_id)

REFERENCES rail\_db.train (train\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS rail\_db.passengerdetails

OWNER to postgres;

CREATE TABLE IF NOT EXISTS rail\_db.station

(

station\_id integer NOT NULL,

name character(50) COLLATE pg\_catalog."default",

arrival\_time time without time zone,

train\_id integer,

hault character(30) COLLATE pg\_catalog."default",

CONSTRAINT station\_pkey PRIMARY KEY (station\_id),

CONSTRAINT train\_id FOREIGN KEY (train\_id)

REFERENCES rail\_db.train (train\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS rail\_db.station

OWNER to postgres;

CREATE TABLE IF NOT EXISTS rail\_db.ticket

(

ticket\_id integer NOT NULL,

passenger\_id integer,

status character(30) COLLATE pg\_catalog."default",

amount integer,

train\_id integer,

CONSTRAINT ticket\_pkey PRIMARY KEY (ticket\_id),

CONSTRAINT train\_id FOREIGN KEY (train\_id)

REFERENCES rail\_db.train (train\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS rail\_db.ticket

OWNER to postgres;

CREATE TABLE IF NOT EXISTS rail\_db.ticketcollector

(

tc\_id integer NOT NULL,

first\_name character(30) COLLATE pg\_catalog."default",

last\_name character(30) COLLATE pg\_catalog."default",

age integer,

CONSTRAINT ticketcollector\_pkey PRIMARY KEY (tc\_id)

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS rail\_db.ticketcollector

OWNER to postgres;

CREATE TABLE IF NOT EXISTS rail\_db.train

(

train\_id integer NOT NULL,

train\_name character(50) COLLATE pg\_catalog."default",

arrival\_time time without time zone,

departure\_time time without time zone,

seats\_available integer,

date date,

tc\_id integer,

CONSTRAINT train\_pkey PRIMARY KEY (train\_id),

CONSTRAINT tc\_id FOREIGN KEY (tc\_id)

REFERENCES rail\_db.ticketcollector (tc\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS rail\_db.train

OWNER to postgres;