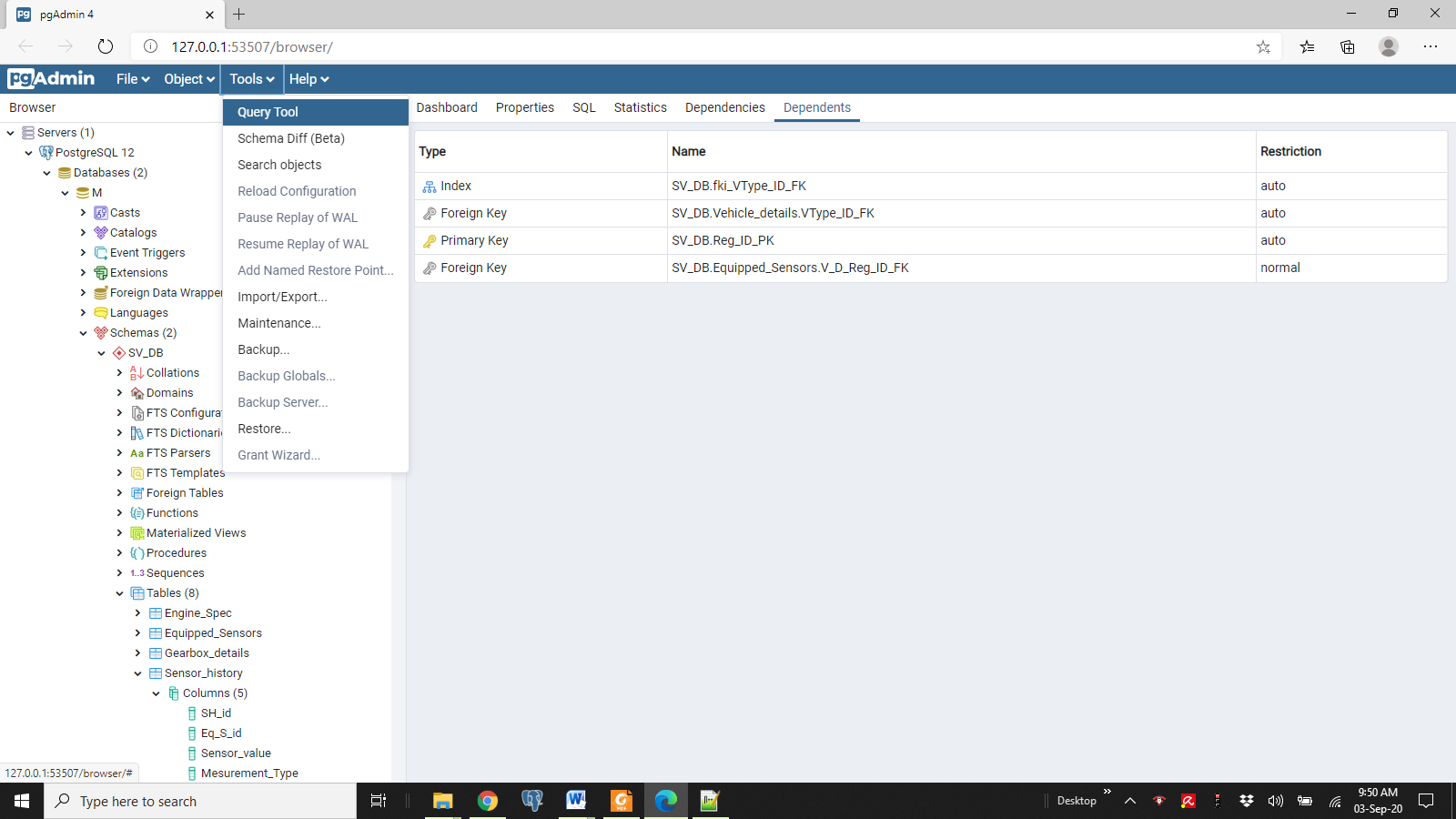
**Objectives**: I) Create tables using below given schema & DDL Scripts in Appendix A.

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

III) Run simple queries.

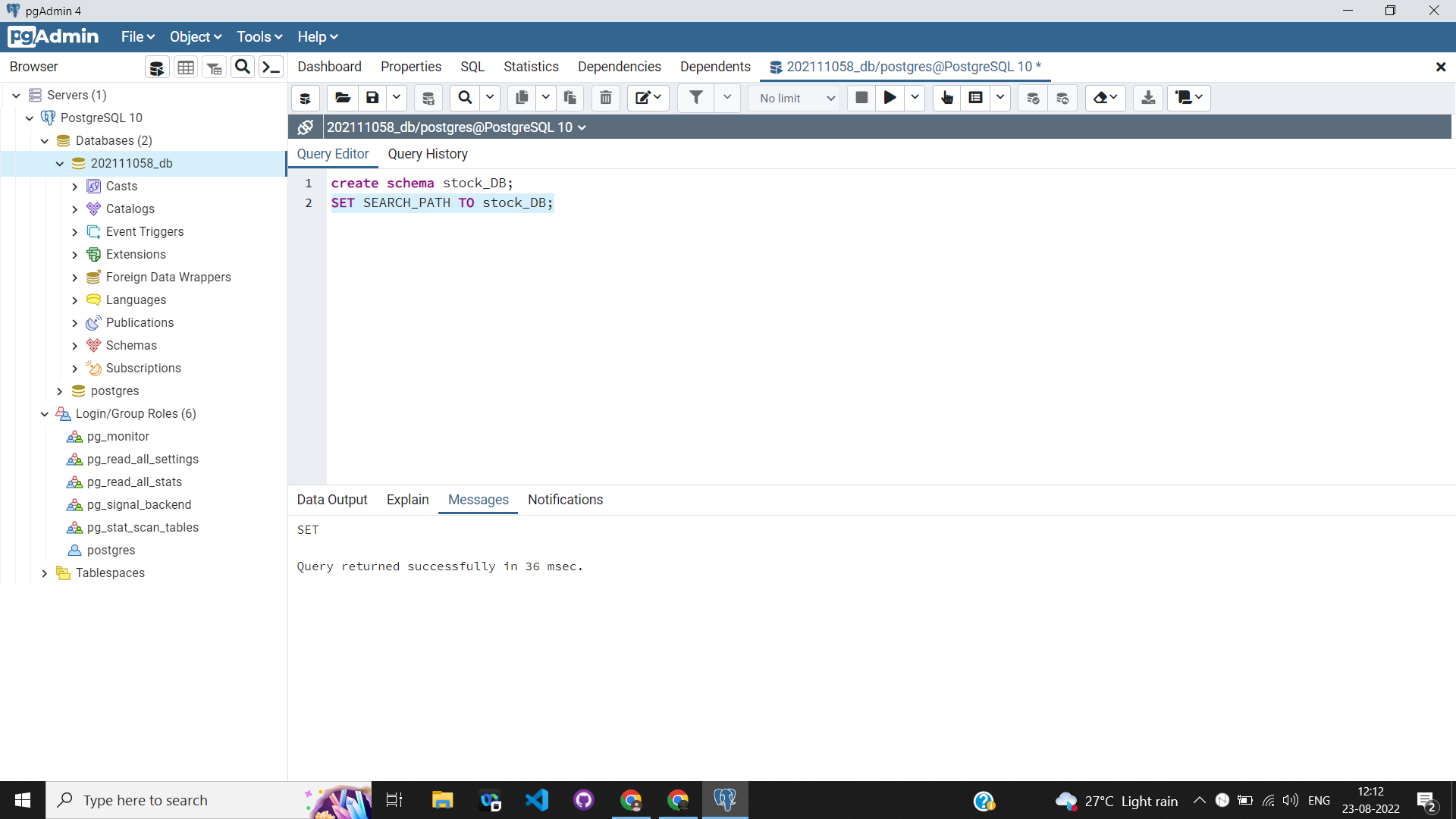
**Submission**: Each student team need to Upload a **single .pdf** file which will contain following things for all the queries listed in (III).

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.

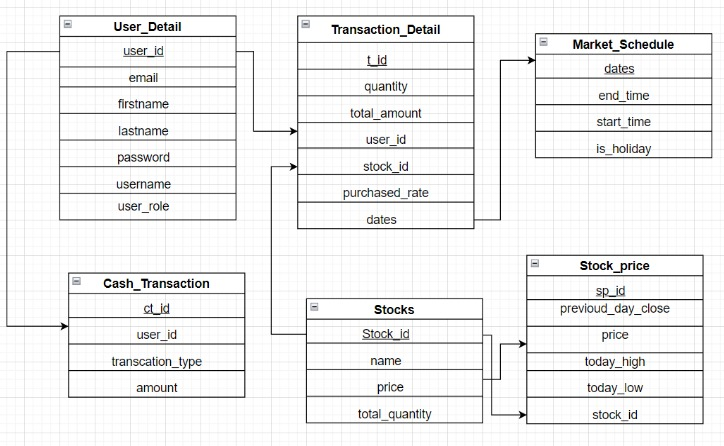


* 1. Type following to create Schema. –

**CREATE SCHEMA stock\_DB;  
SET SEARCH\_PATH TO stock\_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.**

**COPY** stock\_DB.User(user\_id,email,firstname,lastname,username,password,user\_role) FROM 'D:/lab1\_dbms/user.csv' DELIMITER ',' CSV HEADER;

1. **Run queries.**
2. Show all user details.
3. List the names of stocks where the stock price is greater than 50$.
4. Show all the dates where there is no holiday.
5. Count the number of transactions where the amount is less than 300.
6. Find out average stock price.
7. Find out the highest stock price details.
8. Count all rows in market\_schedule
9. List the names of stocks where the total quantity of stocks is less than 100.
10. Find out transactions whose transaction\_type is credit
11. Find the stock details where the today's high is more than 100$.
12. find out stocks where price is greater than 500$ and quantity is greater than 50
13. Count the stock\_id's where the pre close is greater then the today's high
14. Find out the detail of user whose first name is.paul
15. find the details of the cash transaction where the amount is between 100-200$.
16. Display record for the maximum total amount in transaction
17. Show admin’s details
18. Find username of user in ascending order of user\_id for admin
19. Select details of top 2 highest stock price from stocks
20. Print count of different types of cash transactions
21. Print stock id and name whose price is greater than average of price value
22. Find stock names an ids whose price is greater than previous close
23. Find total quantity of stocks date wise whose transaction is done on particular date
24. Find stock name and price whose price is less than average price of today\_low
25. Show table of user sorted email-wise
26. create view of maximum price
27. create a view for admin details then insert and display some new admin to the recently created view.
28. Create and display a view all the details of transactions whose type is debit (debit = false)
29. Display the sum of total amount transaction wise
30. Create a view with transaction made whose quantity is greater than average of all quantity in descending order of amount
31. Display all the transaction details done on holiday

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

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

CREATE TABLE IF NOT EXISTS stock\_db."user"

(

user\_id integer NOT NULL,

email character varying COLLATE pg\_catalog."default",

firstname character varying COLLATE pg\_catalog."default",

lastname character varying COLLATE pg\_catalog."default",

username character varying COLLATE pg\_catalog."default",

password bigint,

user\_role boolean,

CONSTRAINT user\_pkey PRIMARY KEY (user\_id)

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS stock\_db."user"

OWNER to postgres;

—---------------------------------------------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS stock\_db.stocks

(

stock\_id integer NOT NULL,

name character varying COLLATE pg\_catalog."default",

price double precision,

quantity bigint,

CONSTRAINT stocks\_pkey PRIMARY KEY (stock\_id)

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS stock\_db.stocks

OWNER to postgres;

—---------------------------------------------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS stock\_db.market\_schedule

(

dates date NOT NULL,

start\_time time without time zone,

end\_time time without time zone,

is\_holiday boolean,

CONSTRAINT market\_schedule\_pkey PRIMARY KEY (dates)

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS stock\_db.market\_schedule

OWNER to postgres;

—---------------------------------------------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS stock\_db.cash\_transaction

(

ct\_id bigint NOT NULL,

amount integer,

transaction\_type boolean,

user\_id integer,

CONSTRAINT cash\_transaction\_pkey PRIMARY KEY (ct\_id),

CONSTRAINT "ct\_user-id" FOREIGN KEY (user\_id)

REFERENCES stock\_db."user" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS stock\_db.cash\_transaction

OWNER to postgres;

—---------------------------------------------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS stock\_db.stock\_price

(

sp\_id bigint NOT NULL,

pre\_close double precision,

price double precision,

today\_high double precision,

today\_low double precision,

stock\_id bigint,

CONSTRAINT stock\_price\_pkey PRIMARY KEY (sp\_id),

CONSTRAINT "stocks-stock\_id" FOREIGN KEY (stock\_id)

REFERENCES stock\_db.stocks (stock\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS stock\_db.stock\_price

OWNER to postgres;

—---------------------------------------------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS stock\_db.transaction\_detail

(

t\_id bigint NOT NULL,

quantity integer,

purchased\_rate double precision,

total\_amount numeric,

stock\_id bigint,

user\_id bigint,

date date,

CONSTRAINT transaction\_detail\_pkey PRIMARY KEY (t\_id),

CONSTRAINT "market-dates" FOREIGN KEY (date)

REFERENCES stock\_db.market\_schedule (dates) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT "stocks-stock\_id" FOREIGN KEY (stock\_id)

REFERENCES stock\_db.stocks (stock\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT "user-user\_id" FOREIGN KEY (user\_id)

REFERENCES stock\_db."user" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

)

TABLESPACE pg\_default;