Dhirubhai Ambani
Institute of Information and Communication Technology

TITLE: Game Zone

Group ID: S5 Team 7

Subject: Database Management System (DBMS)

Subject code: IT214

Mentor TA: Shreyas Sir

Year: 2021

Group Details:

Student ID	Names
201901014	Harshal Shah
201901074	Dev Vora
201901163	Radhika Patel
201901188	Nisarg Nampurkar

INDEX

Section1: Final version of SRS.

1.Introduction

1.1 Purpose

- To efficiently update, erase, retrieve various customers' records like personal details, membership details, number of points earned, etc. in the scenario where many customers visit the game zone on a daily basis.
- To avoid any kind of potential disputes.
- For easy retrieval of customer details when any unforeseen incident happens.
- Authentication purpose, so that certain privileges are available to only those who have taken membership.
- Helps the administrator and instructor to keep track of the detailed information of the Game zone.
- Assists in the smooth interaction between different types of users(gamers).
- Proper maintenance of available Resources.
- Speed up the Activities to overcome the problems associated with the game zone.

1.2 Intended Audience and Reading Suggestions

- The audience going to play games online/offline is assumed to have basic knowledge of accessing a website or game zone equipment, surfing and knowing about its controls.
- Developers who can review a project's capabilities and more easily understand where their efforts should be targeted to improve or add more features to it.
- Game zone staff and owner.

1.3 Product Scope

The purpose of the online and offline game zone management system is to ease game zone / online gaming management and to create a convenient and easy-to-use place and application for customers, trying to buy balance in a debit card (provided by game zone only) or in an online pocket . The system is based on a relational database with its game zone management and reservation functions. We will have a database server supporting hundreds of major games as well as thousands of games by various gaming companies. Above all, we hope to provide a comfortable user experience along with the best pricing available.

1.4 Description

In a Game Zone, customers register at our system and get a debit card from the game zone.A game zone database system stores the following information.

- **Game details:**

  It includes various game details,along with the game price, eligibility criteria, waiting time etc.

- **Customer description:**

  It includes the customer name, address,height,weight and phone number. This information may be used for keeping the records of the customer for any other kind of information.

- **Credit card description:**

  It includes customer details, card number, date of buying, date of expiry, balance, payment history, authentication of user.

1.5 Description (Extended)

In a Game Zone, customers register at our system and get a debit card from the game zone.A game zone database system stores the following information. There is a database for customers with all records, playing games, register for that game, price of a game, eligible age/height, timing, waiting list etc in real-time with history for each game. For certain games winners may be warded prices. Different teams can take up different types of games including online & offline games which need additional equipment.

- **Game details:**

  It includes various game details,along with the game price, eligibility criteria, waiting time etc.

  - When a manager wants to add a game, the facility of adding a new game is also given. Manager must add the new game's price, eligibility criteria, waiting time etc details.
  - In certain situations, if the manager wants to delete games, this facility is also given.
  - If a customer wants to know certain game details, he/she can access game details. Customers can only see game details, modification,insertion or deletion is not allowed.

- **Customer description:**

  It includes the customer name, address,height,weight and phone number.

  - This information may be used for keeping the records of the customer for any other kind of information.
  - It includes a dynamic list of customers visiting the game zone.
  - Whenever the customer will come and its details are not available in the database, then staff will add new customer details.
  - Also, safety is to be ensured by keeping record of vaccination status of customers.

- **Credit card description:**

  It includes customer details, card number, date of buying, date of expiry, balance, payment history, authentication of user.

    - When a new customer comes, a new credit card will be given to him/her. Money will be added to the customer's credit card.
    - Also existing customer’s credit card validity will be checked when he/she visits the game zone.
    - When a customer plays any game, credit card balance will be modified.When the customer’s card balance is less than the required balance for the game, he/she can’t play that game.
- **Reward system description**
    - To attract customers, exciting gifts for different games are given to customers who score the required points.
    - Information of the gift's id, name, and required points to get that gift is stored in the database.
    - Also, the information of available stock of the gift is included.
- **Game playing history**
    - To be updated each time a game is accessed by a customer, with proper details like time, the game, points won, any damage caused, malpractices/cheating,etc.
- **Transaction Episode**
    - The corresponding table needs to be updated with any payment transaction, reward transaction, etc. Also, the changes should be reflected in the database in a manner that integrity is maintained.
- **Staff Details**
    - We include a table of staff which contains information about them like the name of the staff member, age, phone number, their ongoing health issues, blood group, date of joining, etc. This information helps the owner to keep track of staff.
- **High score information**
    - By this entity set, we can keep track of the highest score achieved in each game and details of who earned that score.These details will push customers to score more.

2. Requirements collection/Fact finding phase

2.1 Background Reading/s

References :

A. iCafeCloud

- Link to the original work
- Welcome to iCafeCloud - All in one Cafe Billing and Game Management Software
- iCafeCloud is one such platform which is used to manage game zone, cyber cafe-like environment by giving a database management facility. It is a proprietary software
- iCafeCloud is a simple billing and game management program to manage customers accounts and billing for cafe users as well as game updates and license management. Each day the game zone has hundreds of users and game updates, keeping track of them all can be hard. So, iCafeCloud simplified the task.

- Facilities offered:
    - Keeps track of members
    - Generate reports
    - Give user more value by giving bonuses and prizes
    - Manage your PC'S and Consoles by data storage facility ICafeCloud functions allow you to turn on/off PC's and devices when user sessions are started and finished. Allowing you to focus on making money.
    - System to rewards prize

- Functionalities and privileges- How the above facilities are offered?(Working flow, features)
- Given below are the functionalities, features provided by the platform to implement the above facilities
    - iCafeCloud: Server installation
    - iCafeCloud: Client installation
    - iCafeCloud: Server settings
    - iCafeCloud: How to download and add games
    - iCafeCloud: How to add product and restock
    - iCafeCloud: How to cancel orders on shop
    - iCafeCloud: How to place orders from iCafeCloud client PC
    - iCafeCloud: How to set full screen mode | iCafeCloud client
    - iCafeCloud: How to add license to license pool
    - iCafeCloud: How to create offers and add offer to members
    - iCafeCloud: Set price for PC group and member group
    - iCafeCloud: How to set price buttons
    - iCafeCloud: How to use bonus function
    - iCafeCloud: How to create guest account
    - iCafeCloud: How to use points function
    - iCafeCloud: How to add members
    - iCafeCloud: How to book PC time
    - iCafeCloud: How to add local steam game
- Flaws
    - The game update takes too much time.
    - Lack of privacy and security of the functional system, certain privileges related to user relation should be evoked from staff.

- The software is quite costly.

B.
Cyber Cafe Management System Using PHP & MySQL , Cyber Cafe Management System Project (phpgurukul.com)

- **Query language used:** MySQL
- In this Cyber Café Management System, PHP and Mysql databases are used.
- This is the project which keeps records of daily users of cyber cafes.
- **Cyber Café Management System has one module i.e admin.**
- **Features provided include:**
    1. **Dashboard**
    2. **Computer**
    3. **Users**
    4. **Search**
    5. **Report**
    6. **Profile**
    7. **Change Password**
    8. **Logout**
    9. **Admin can also recover his/her password.**
- **Note**: In this project MD5 encryption method is used.
- Flaws
    - System maintenance is too poor.
    - Too less functionalities available.
    - Low quality of system equipment, technology used.
    - Users' privacy and security is taken less seriously.
    - Not robust enough to stand technical errors.

- List the combined Requirements gathered from Background Reading/s.
    - Need for a system that can handle concurrent access in a better way.
    - A system that is immune to sudden power cuts, should atomically execute requests, transitions and be fair to diverse classes of users.
    - The need of handling privilege distribution more carefully to better preserve customers' privacy and security.
    - Systems should be more generalized for easy maintenance.
    - Latest technology to better handle transactions at low cost.
    - Functionalities to be increased.
    - Time complexity can be made better at software level.

2.2 Interviews:

1) Owner's interview

- Interview plan

TIMES Game Zone: (Role Play/Actual) Interview Plan

System: database management system for game zone

Project Reference: SF/SJ/2021/12

Interviewee: 1) Ghanshyam Shah **(Role Play/Actual)**
Designation: Owner at TIMES
Contact Details: Email/Phone no. **(Only for Actual)**
Organization Details: Name & Address **(Only for Actual)**

Interviewer: 1) Radhika Patel **Designation**: Software developer at TIMES

Date:4/10/2021 **Time:** 14:00

Duration: Max 45 min **Place:** Radhika's office

Purpose of Interview:
Preliminary meeting to identify problems and requirements of database management system for game zone

Agenda:
Problems with security and any other concerns
Privileges hold by Owner of the game zone

Documents to be brought to the interview:
None

- Interview summary

TIMES game zone: Mock Interview Summary

System: database management system for game zone

Project Reference: SF/SJ/2003/12

Interviewee: 1) Ghanshyam Shah**(Role Play)**
Designation: Owner of game zone

Interviewer: 1) Radhika Patel **Designation**: Software developer at TIMES

Date:4/10/2021 **Time:** 14:00

Duration: 20 minutes **Place:** Radhika's office

Purpose of Interview:
Preliminary meeting to identify problems and requirements of database management system for game zone

1. Some security issues like users can change other users information
2. System crash when too many users want to access game
3. Difficult to know which game is more profitable
4. Don't know which games are most favorite in users(no track of rating of games)

2) Manager's interview

- Interview plan

TIMES Game Zone: (Role Play/Actual) Interview Plan

System: database management system for game zone

Project Reference: SF/SJ/2003/12

Interviewee: 1) Sandeep Patel **(Role Play)**
Designation: manager at game zone

Interviewer: 1) Dev Vora
Designation: Software developer at TIMES

Date:5/10/2021 **Time:** 11:00

Duration: Max 45 min **Place:** Dev's office

Purpose of Interview:
Preliminary meeting to identify problems at management level of database and requirements from the management team

Agenda:
Problems in management of database
Important requirements

Documents to be brought to the interview:
None

- Interview summary

TIMES game zone: Mock Interview Summary

System: database management system for game zone

Project Reference: SF/SJ/2003/12

Interviewee: 1) Sandeep Patel**(Role Play)**
Designation: manager at game zone

Interviewer: 1) Dev Vora
Designation: Software developer at TIMES

Date:5/10/2021 **Time:** 11:00

Duration: 20 minutes **Place:** Dev' office

Purpose of Interview:
Preliminary meeting to identify problems at management level of database and requirements from the management team

1. No feature to check validity of credit card
2. How to remove game which is not much profitable. That functionality is not provided.
3. No feature to add a game when users ask for particular game many times.

3) Operating staff's interview

- Interview Plan

TIMES Game Zone: (Role Play/Actual) Interview Plan

System: database management system for game zone

Project Reference: SF/SJ/2003/12

Interviewee: 1) Vraj Vora (**Role Play**)
Designation: operator of some games in game zone
2) Pratham nampurkar (Role play)
Designation: operator of some games in game zone

Interviewer: 1) Harshal Shah
Designation: Software developer at TIMES

Date:4/10/2021 **Time:** 15:00

Duration: Max 45 min **Place:** Harshal's office

Purpose of Interview:
Preliminary meeting to identify problems in the current database and additional requirements of the operating staff

Agenda:
Problems faced by operating staff
Requirements of operating staff

Documents to be brought to the interview:
None

- Interview summary

TIMES game zone: Mock Interview Summary

System: database management system for game zone

Project Reference: SF/SJ/2003/12

Interviewee: 1) Vraj Vora (**Role Play**)
Designation: operator of some games in game zone
2) Pratham Nampurkar (Role play)
Designation: operator of some games in game zone

Interviewer: 1) Harshal Shah
Designation: Software developer at TIMES

Date:4/10/2021 **Time:** 15:00

Duration: 25 minutes **Place:** Harshal' office

Purpose of Interview:
Preliminary meeting to identify problems in the current database and additional requirements of the operating staff

1. Updating the winners of game whenever someone plays that game takes long amount of time
2. There is no queue limit and maximum waiting time for a game
3. Database does not have information about user's height, weight. So, it's difficult to decide whom to allow to play a game

List of requirements :

- System should be able to handle crash and operation failures during certain task

- Rating of the game should be included and updates continuously, so we can know which games are popular
- feature to add a new game, when customers demands it
- Updating the highest score and winner of the game should be done on a daily basis.So customers who come in the same week can get correct information about the game.
- Information about the customer's height, weight should be included. So staff can decide if customers should be allowed to play a game or not.
- Concept of queue limit and maximum waiting time
- Validity of credit card should be included.
- There should be a feature to remove a game which has not been used for a long time and is not very profitable.
- Privacy and security of the database should be maintained

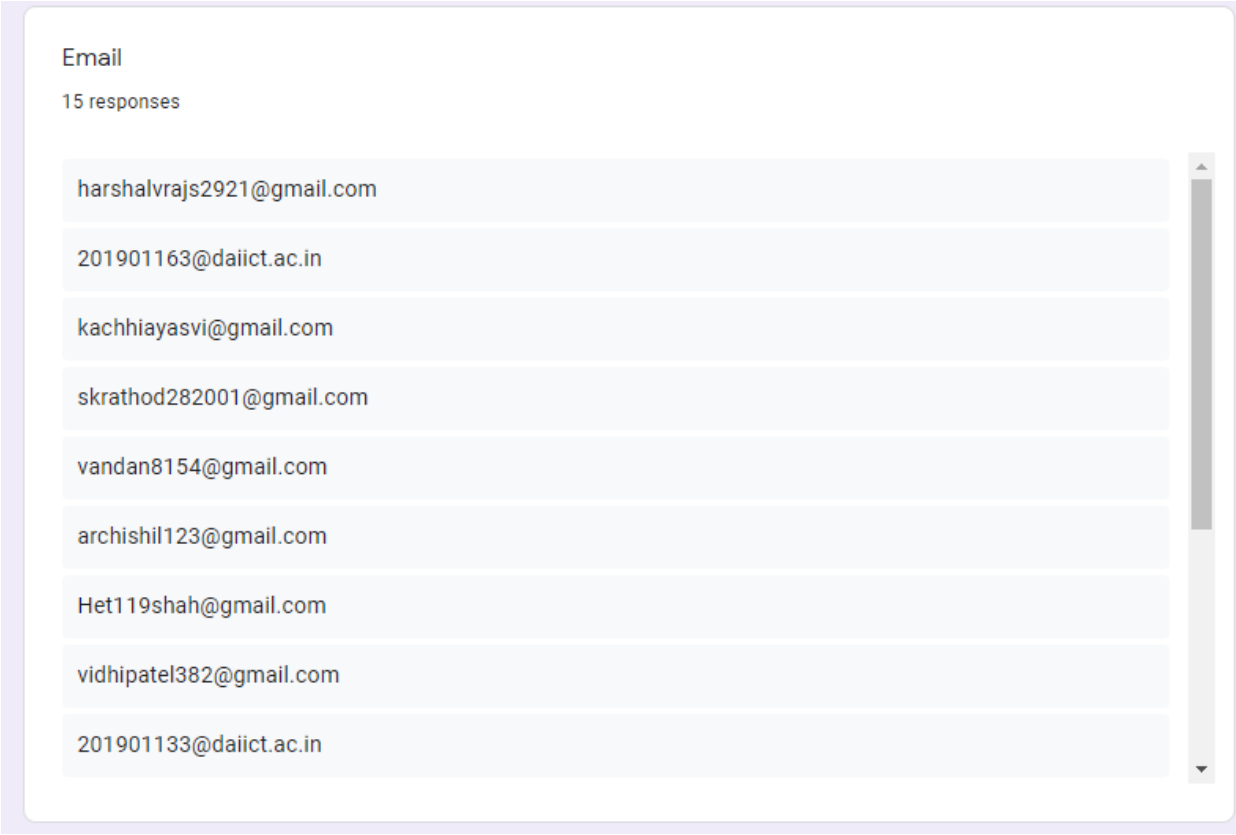
2.3 Questionnaires

- For manager and staff of game zone

Google form for staff and manager

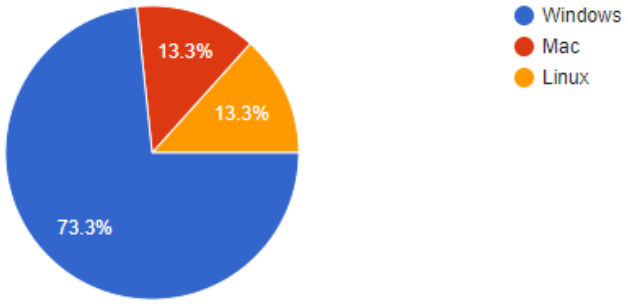
Excel Sheet

Summary:



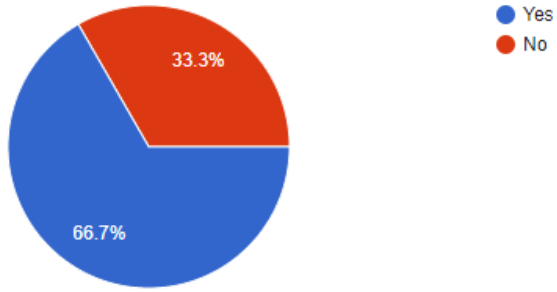
What operating system you are using?

15 responses



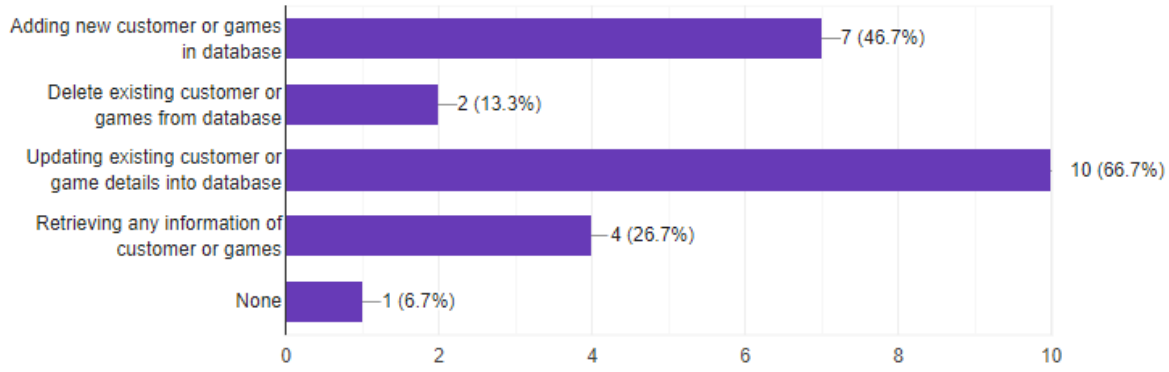
Have you often facing trouble with your system?

15 responses



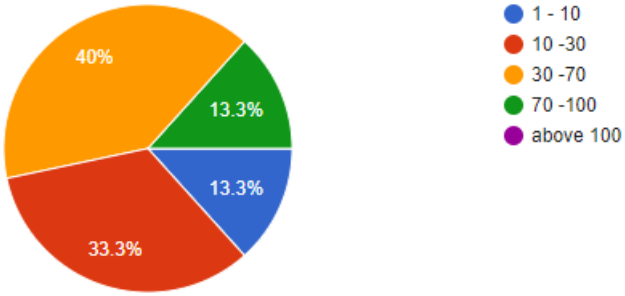
what operations make you more trouble?

15 responses



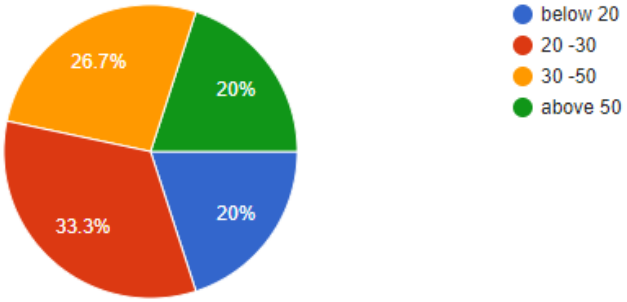
How many customers come to your Game zone on Daily bases?

15 responses



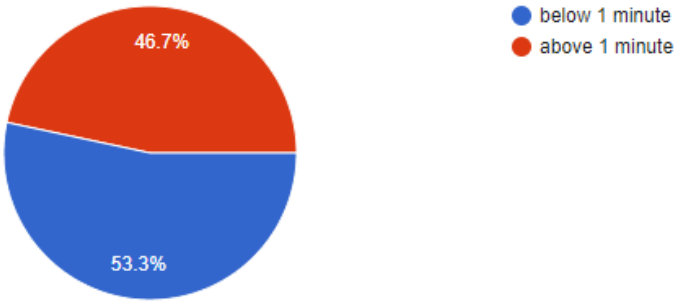
How many games in your Game Zone?

15 responses



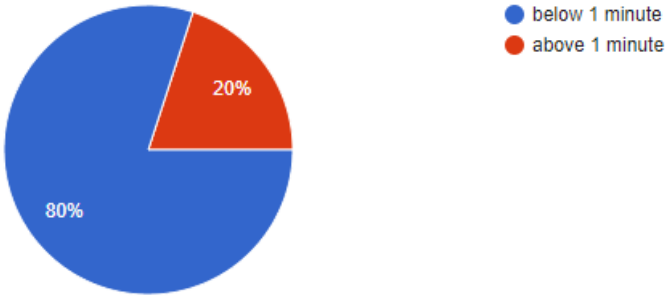
How long it takes to insert new customer details or game details in system ?

15 responses



How long it takes to update customer details or game details in system ?

15 responses



What additional fields (in any section) you require for your system ?

10 responses

- Mobile number of customer
- Ratings for game
- Partner
- Nothing
- -
- Age of customer
- Customer Rating
- Game Rating
- Field which shows current balance of customer

Give any comments or suggestion on existing Database system.

9 responses

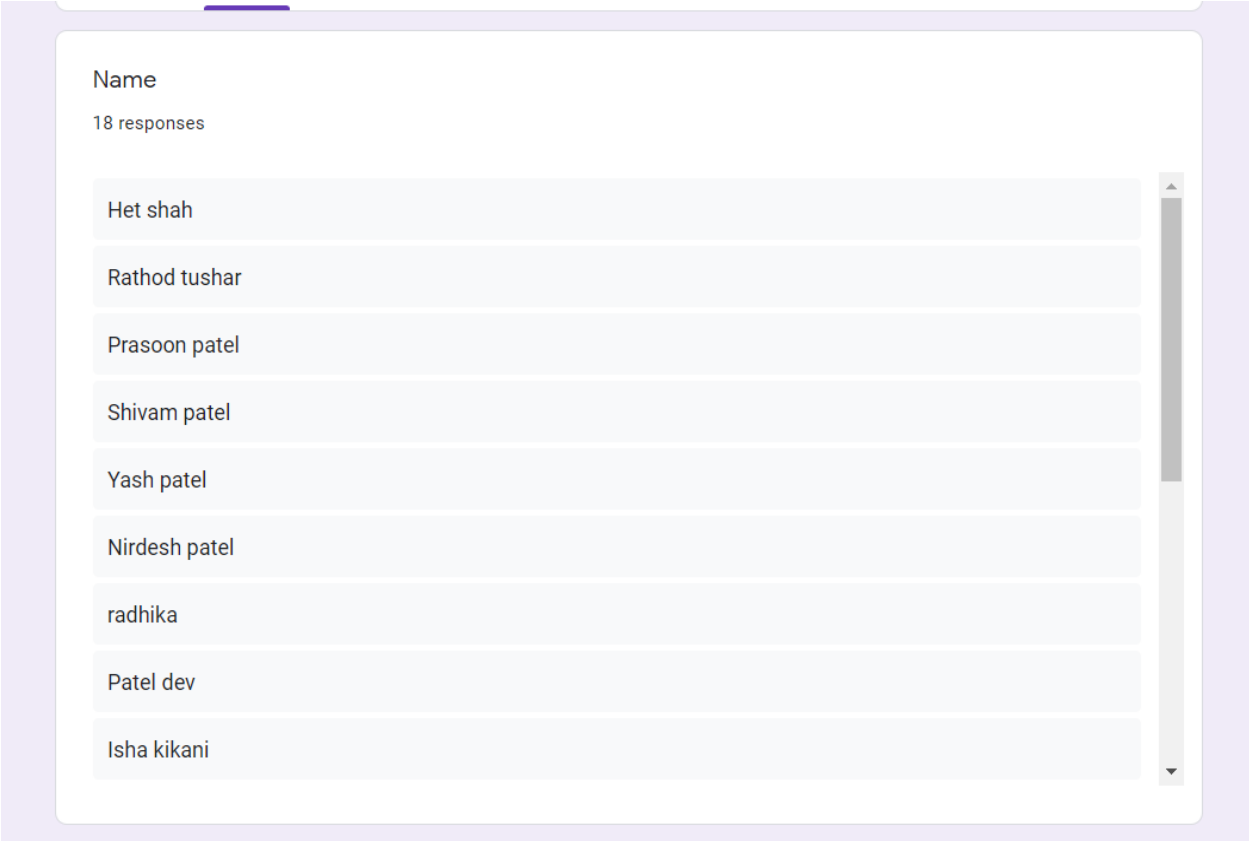
- Excellent
- -
- No feature to know which game is more profitable, which game is most popular
  So i need game rating
- Updating high scores and winners of the game is done on a weekly or monthly basis. So it is difficult to know high correct game information for the customer who comes in same week or month so it should be of daily basis
- Feature of addition of new game which is most demanded or removal of game which is not in demand this opration is provided very slow it should faster.
- How to decide credit card which is issued to customer is valid or not so it must contain unique card id
- We give customer certain credit point on their every new filling up balance so it should add to their main balance when they need and this is very nice statargy to attract customer

- For customers

[Google form for customers]

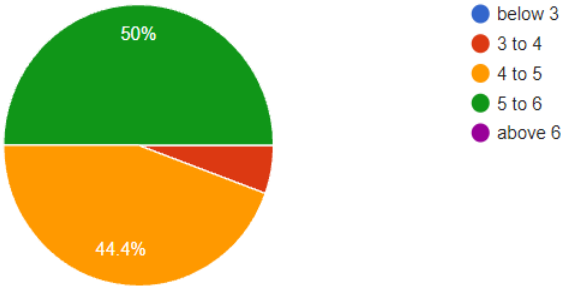
[Excel sheet of responses]

Summary:



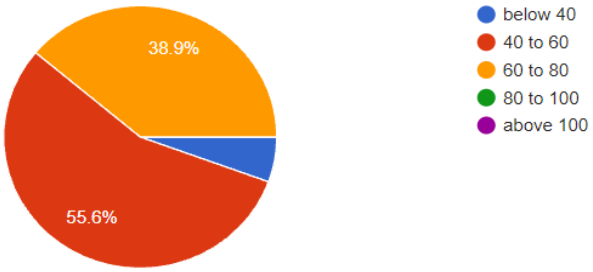
1. what is your height(in feet)?

18 responses



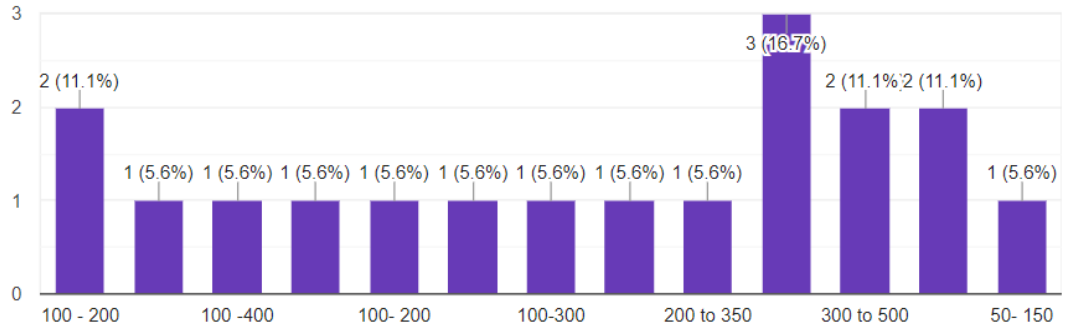
2. What is your weight(in Kg)?

18 responses



3. What is your budget range for different games?

18 responses



Which game would you like to play? Give any comments from your experience.

18 responses

- Glow hockey, bowling , car race
- Trempling, bike race
- VR games, counter strike
- VR games, trempling , table football
- Redemption games
- Bike race , jumping games
- glow hockey, car racing
- Air hockey,all shooting games
- Table tennis, counter strike, chess

Combined Requirements :

- Please make sure that it takes less time to insert and update .
- feature to know which game is more profitable, which game is most popular So need for game rating.
- Updating high scores and winners of the game is done on a weekly or monthly basis. So it is difficult to know high correct game information for the customer who comes in the same week or month so it should be on a daily basis .
- Feature of addition of new game which is most demanded or removal of game which is not in demand this operation is provided very slow it should be faster.
- How to decide credit card which is issued to a customer is valid or not so it must contain a unique card id.
- We give customers certain credit points on every new filling up balance so it should add to their main balance when they need it and this is a very nice strategy to attract customers.
- Mobile number of customer
- Ratings for game
- Partner if any game need
- Age of customer
- Customer Rating
- Game Rating
- Field which shows current balance of customer
- Customer earn point detail

2.4 Observations:

System : Gamezone Management System
Observations by : Vora Dev (Gamezone System - Owner)
Date : 08/10/2021 **Time:** 16:54
Duration: 45 mins

- Electricity fluctuations are very frequent. Urgent requirement of generators.
- Lack of privacy and security of the functional system. Easy to hack.
- Poor Updatition system. Delay found in updating user's details.
- Poor maintenance of system used.
- Low quality of system equipment used.
- Long waiting time.
- System not able to handle large number of customers.

3. Fact Finding Chart

No.	Objective	Technique	Subject(s)	Time
1	To study the existing systems if there exist any and to get a precise idea of designing a database for such a case.	Background reading	Cybercafe management system (software)	0.5 day
2	To know the drawbacks in the existing systems if there exists any	Background reading	iCafeCloud (software), Online reviews	0.5 day
3	To know the requirements from owner	Interview	Owner	20 mins
4	To get knowledge of current problems at management level	Interview	Manager	20 Mins
5	To get understanding about what information needed from customer	Interview	Operating Staff (any 2 staff members)	2 x 25 mins each
6	To know major problems faced by various game zone's staff	Questionnaire	All staff members of different game zone	5 mins per staff member
7	To know about privacy and security	observations	Game zone's database	One week
8	To know storage capacity of database	observations	Game zone's database	4 or 5 days
9	To know number of daily customers	Questionnaire	Game zone staff and manager	5 mins per person

4. List Requirements:

- System should be able to handle crash and operation failures during certain task
- Rating of the game should be included and updates continuously, so we can know which games are popular
- feature to add a new game, when customers demands it
- Updating the highest score and winner of the game should be done on a daily basis.So customers who come in the same week can get correct information about the game.
- Information about the customer's height, weight should be included. So staff can decide if customers should be allowed to play a game or not.
- Validity of credit card should be included and it should be unique.
- Privacy and security of the database should be maintained
- Please make sure that it takes less time to insert and update .

- We give customers certain credit points on their every new filling up balance so it should add to their main balance when they need it and this is a very nice strategy to attract customers.
- Mobile number of customer
- Partner if any game need
- Age of customer
- Customer Rating
- Field which shows current balance of customer
- Customer earn point detail
- Systems should be more generalized for easy maintenance and Functionalities to be increased.
- Latest technology to better handle transactions at low cost.
- Time complexity can be made better at software level.

5. User Classes and Characteristics:

Admin:

- Can Add/Delete any game
- Can modify existing game's properties (e.g. price,time limit, max_number of players playing a game simultaneously, game benefits like price discount, winning price.
- Can add/delete/modify card's properties like card price limits, minimum amount for a card, card expiry dates, card types
- Can add/delete/modify staff's properties like income, time shifts, bonuses, uniforms.

Staff:

- Can Add/delete customers
- Can modify customer details like name,phone number,address,height,weight,disabilities,etc.
- Can modify card details of customers like card balance, card type used

6.Operating Environment (for PostgreSQL)

6.1 Hardware, Software or Connectivity Requirements

Hardware Requirements :

There has been a lot of talk through the years of what the minimum set of hardware requirements are for a PostgreSQL database. Generally speaking the requirements for PostgreSQL are very low, you can even get by on 256 Megs of memory. However you rarely hear or read about what would be considered the minimum production hardware requirements for PostgreSQL.

Minimum Production Requirements

- 64 bit CPU
- 64bit Operating System
- 2 Gigabytes of memory
- Dual CPU/Core

Software Requirements :

PostgreSQL and EnterpriseDB Version 9.2 to 13.x

Operating System	**Operating System Version**	**Architecture**
AIX	AIX 7.1 64-bit	PowerPC
FreeBSD	FreeBSD 10.x	· x64 · Compatible processors
		·
	FreeBSD 9.x *Supported for PostgreSQL Version 9.2 only	· x64 · Compatible processors
Linux	**Debian**	
	Debian 9.x	· x64 · Compatible processors

	Debian 8.x	· x64 · Compatible processors
	Debian 7.x	· x64 · Compatible processors
	Debian 6.x	· x64 · Compatible processors
	Debian 5.x	· x64 · Compatible processors
	Red Hat Enterprise Linux/CentOS	
	Red Hat Enterprise Linux/CentOS 8.x with glib 2.28.x Before you install the Commvault software, you must manually install the libxcrypt.x86_64 and libnsl.x86_64 packages on the computer.	· x64 · Compatible processors

	Red Hat Enterprise Linux/Oracle Linux Enterprise/CentOS 7.x	· x64 · PowerPC (little endian) · Compatible processors
	Red Hat Enterprise Linux/Oracle Linux Enterprise/CentOS 6.x with glibc 2.12.x	Intel Pentium, x64, Power PC (Little endian) or compatible processors
	Red Hat Enterprise Linux/Oracle Linux Enterprise/CentOS 5.x with glibc 2.5.x	· Intel Pentium · x64 · Compatible processors
	Source Mage Linux	
	Source Mage Linux	x64
	SuSE Linux (SLES)	
	SuSE Linux 12 Initial Release	· x64 · PowerPC (little endian) · Compatible processors
	SuSE Linux 11 (Initial Release/SP1 and later SPs) with glibc 2.9.x and higher	Intel Pentium, x64, Power PC (Little endian) or compatible processors

	Ubuntu	
	Ubuntu 20.04 LTS	· Intel Pentium · x64 · Compatible processors
	Ubuntu 18.04 LTS	· Intel Pentium · x64 · Compatible processors
	Ubuntu 16.04 LTS	· Intel Pentium · x64 · Compatible processors
	Ubuntu 14.04 LTS	· Intel Pentium · x64 · Compatible processors
	Ubuntu 12.04 LTS	· Intel Pentium · x64 · Compatible processors

Solaris	Solaris 11.x	· x64 · Sparc T/M series
	Solaris 10 Update 6 or equivalent	· x64 · Sparc T/M series
Windows	**Windows Server 2019**	
	Microsoft Windows Server 2019 Editions	All Windows-compatible processors supported
	Windows 2016	
	Microsoft Windows Server 2016 Editions Core Editions are not supported.	All Windows-compatible processors supported
	Windows 2012	
	Microsoft Windows Server 2012 R2 Editions Core Editions are not supported.	All Windows-compatible processors supported
	Microsoft Windows Server 2012 Editions Server Core installations are not supported. Core Editions are not supported.	
	Windows 2008	

	Microsoft Windows Server 2008 R2 Editions Core Editions are not supported.	All Windows-compatible processors supported
	Microsoft Windows Server 2008 Editions Server Core installations are not supported.	All Windows-compatible processors supported

Connectivity Requirements :

- IP connections must exist between the Delphix Engine and the target environments.
- The Delphix Engine uses an **SSH** connection to each target environment, **NFS** connections from each target environment to the Delphix Engine, and **PostgreSQL client** connections to the virtual databases on the target environment.
- Once connected to a staging target environment through **SSH**, the Delphix Engine initiates a **PostgreSQL replication client** connection from the target environment to the source environment.

6.2 External Interface Requirements

Hardware/Software/Third-party APIs & other things taken from other external sources.

There are only two client interfaces included in the base PostgreSQL distribution:

- **libpq** is included because it is the primary C language interface, and because many other client interfaces are built on top of it.
- **ECPG** is included because it depends on the server-side SQL grammar, and is therefore sensitive to changes in PostgreSQL itself.

All other language interfaces are external projects and are distributed separately. **Table H.1** includes a list of some of these projects. Note that some of these packages might not be released under the same license as PostgreSQL. For more information on each language interface, including licensing terms, refer to its website and documentation.

Table H.1. Externally Maintained Client Interfaces

Name	Language	Comments	Website
DBD::Pg	Perl	Perl DBI driver	**https://metacpan.org/release/DBD-Pg/**

JDBC	Java	Type 4 JDBC driver	**https://jdbc.postgresql.org/**
libpqxx	C++	C++ interface	**https://pqxx.org/**
node-postgres	JavaScript	Node.js driver	**https://node-postgres.com/**
Npgsql	.NET	.NET data provider	**https://www.npgsql.org/**
pgtcl	Tcl		**https://github.com/flightaware/Pgtcl**
pgtclng	Tcl		**http://sourceforge.net/projects/pgtclng/**
pq	Go	Pure Go driver for Go's database/sql	**https://github.com/lib/pq**
psqlODBC	ODBC	ODBC driver	**https://odbc.postgresql.org/**
psycopg	Python	DB API 2.0-compliant	**https://www.psycopg.org/**

7. Product Functions

- Add game
    - Add game function will take input game name, price, rating etc.

- Delete game
    - Delete game function will take the input game name and delete all the game details.

- Modifying game details
    - This function will take input game name and corresponding new information we want to update.

- Add/delete/ Modify card details
    - add card details function will take details like- card number, card balance, name of the card holder etc.
    - delete card details function will take details of card number and delete corresponding card information.
    - The Modify card details function will take details like card number and corresponding information we want to modify.

- Add/delete/Modify staff details
- Add/ Delete customer details
    - Add function will take inputs like customer id, name,height,weight, credit card number etc.
    - Delete function will take input of the customer id and name.
    - Modify function will take input of customer id and corresponding information we want to modify.

- Add a payment/reward transaction
    - Whenever a customer takes a membership(purchases card) or pays for a game, or claims a reward in lue of his tokens/coins earned, there is an event of transaction.
    - The details of the transaction episode are stored in the database.

8. Privileges

Functions	**Privileged users**
Add game	Admin
Delete game	Admin
Modify game details	Admin
Add/delete/modify card details	Admin, staff
Add/delete/modify staff details	Admin
Add/delete customer details	Admin, staff
Add a payment/reward transaction	Admin, staff

9. Assumptions

- Customer details modification doesn’t take too much time.
- Modification of game details is instantaneous.
- Card balance modifications don't take much time.
- Up to date system equipment used.
- Only the admin has access to game modification functions.
- The customer's data privacy is maintained.
- The user knows how to operate computers.
- The user knows the English language.
- The user has reliable Internet connection.

10. Business constraints

- An old user must have the card (digital/hardcopy) given to him otherwise he/she will have to consider purchasing a new one.
- The information of all the existing users must be stored appropriately in the database which can be accessed by Admin/Staff.
- The interface is provided in the English language only. Hence users must know how to speak/read/type English.
- The user must be able to access the system from any computer with proper internet functionalities and capabilities.

Section2: Noun Analysis.

1. Noun(& verb) Analysis

1.1 Extracted noun and verbs

Noun	**Verbs**
customers	credit
games	card
description	form
rating	Register
gamezone	management
details	includes
price	game
eligibility	gameplay
information	play
number	includes
card	customer
date	management
credit	customer
balance	get
name	following
age	pay
height	add
weight	update
phone_number	modify
Gaming history	delete

expiry	record
criteria	authentication
address	store
any	waiting
other	buying
kind	used
various	buying
time	warded
user	winning
id	access
player	event
gifts	visits
eligible	vaccination
tournament	transfer
Counter-Strike	attracting
manager	available
dynamic	manages
staff	score
bank	calculation
reflection	earned
integrity	joining
Blood	

Group	
health	
stocks	
points	

1.2 Accepted Noun and Verb List

Candidate entity set	**Candidate attribute set**	**Candidate relationship set**
Customer	Customer_id Weight Height	Gaming History Customer_payment_info High Scorer
Game	Game_id Game_name Price No_of_player Minimum_height Minimum_weight rating	Gaming History Manages Rewards High_Score calculation
Credit Card	Credit_Card_no Balance Buying_date Expiry_date Total_points	Customer_payment_info
Gifts	Product_id Product_name required_points Avai	Rewards
Person	Phone Number Name Age	
Staff	Staff id Salary Date of Joining Blood Group Health_issue	Manages

High Score	Score (Points_scored) Date	High_Score Calculation High Scorer

1.3 Rejected Nouns

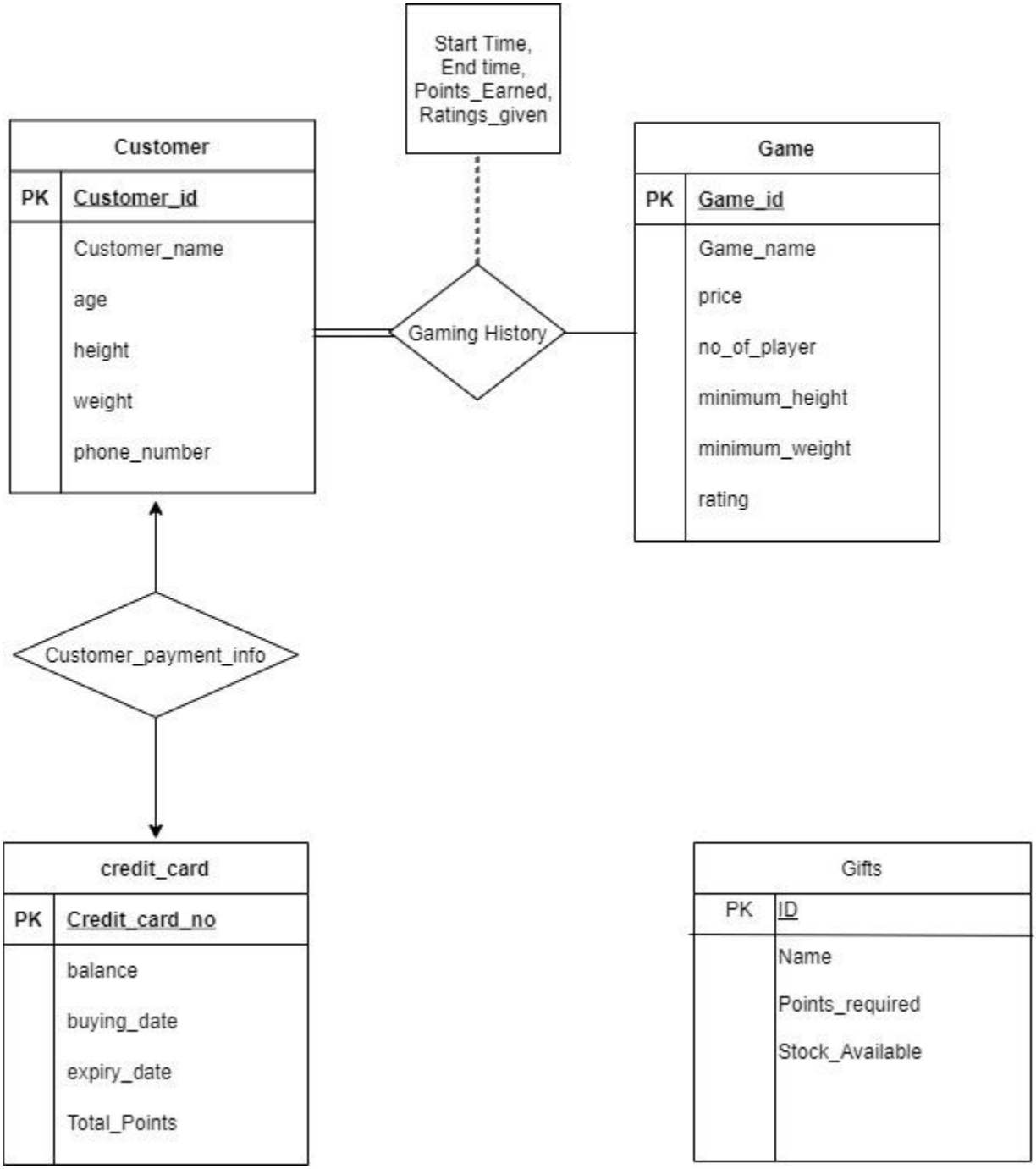
Nouns	**Rejected Reason for Nouns**
gifts	General
time	Vague
user	duplicates
various	Irrelevant
kind	Irrelevant
other	Irrelevant
any	Irrelevant
criteria	Vague
credit	General
date	General
card	General
number	General
information	Irrelevant
eligibility	Vague
details	Vague
gamezone	Vague
description	Vague
eligible	Vague
tournament	General
Counter-Strike	Irrelevant
manager	Vague
dynamic	Irrelevant
Staff	Vague

bank	Irrelevant
reflection	Irrelevant
integrity	Irrelevant

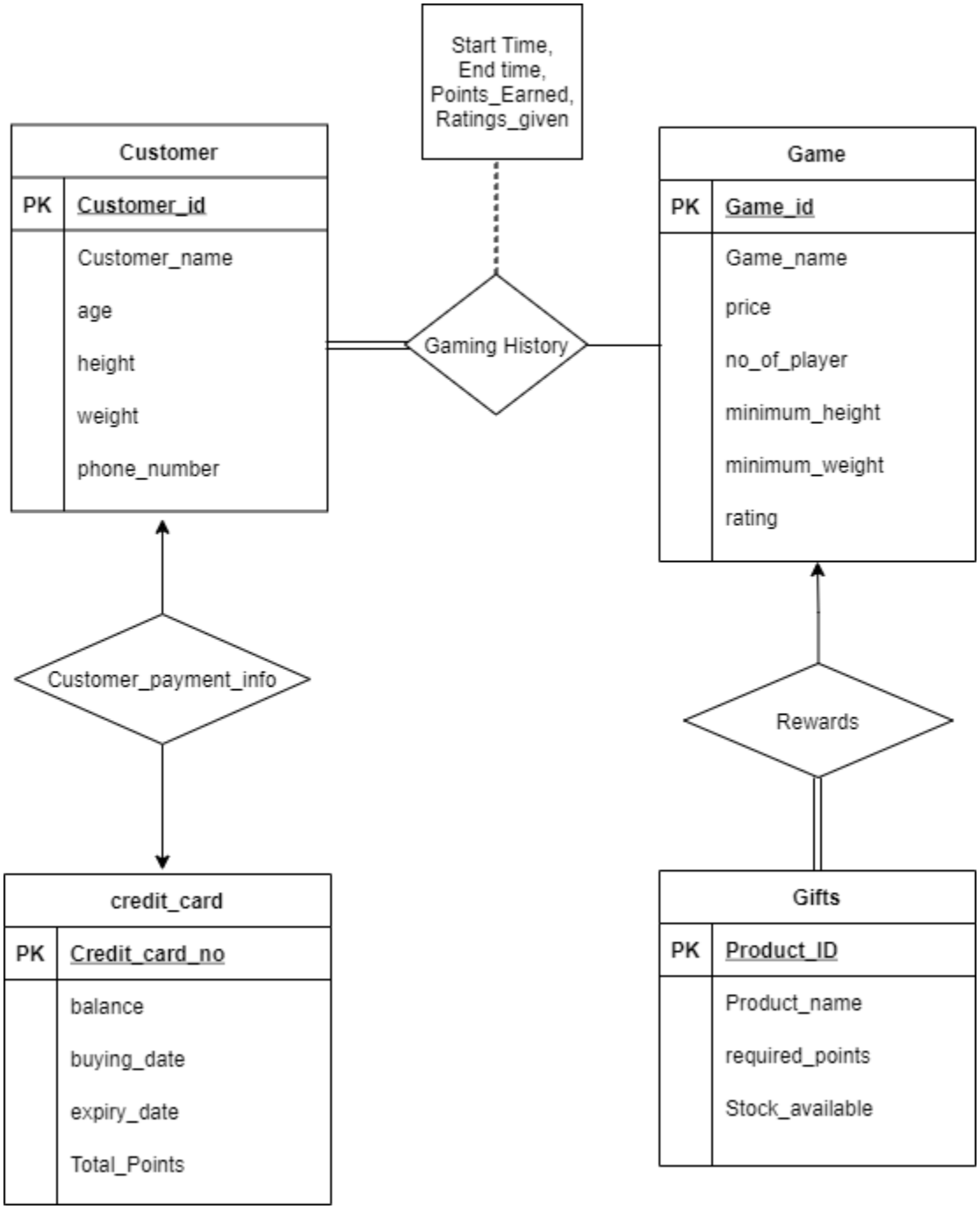
Section3: ER-Diagrams all versions

ER Diagram:

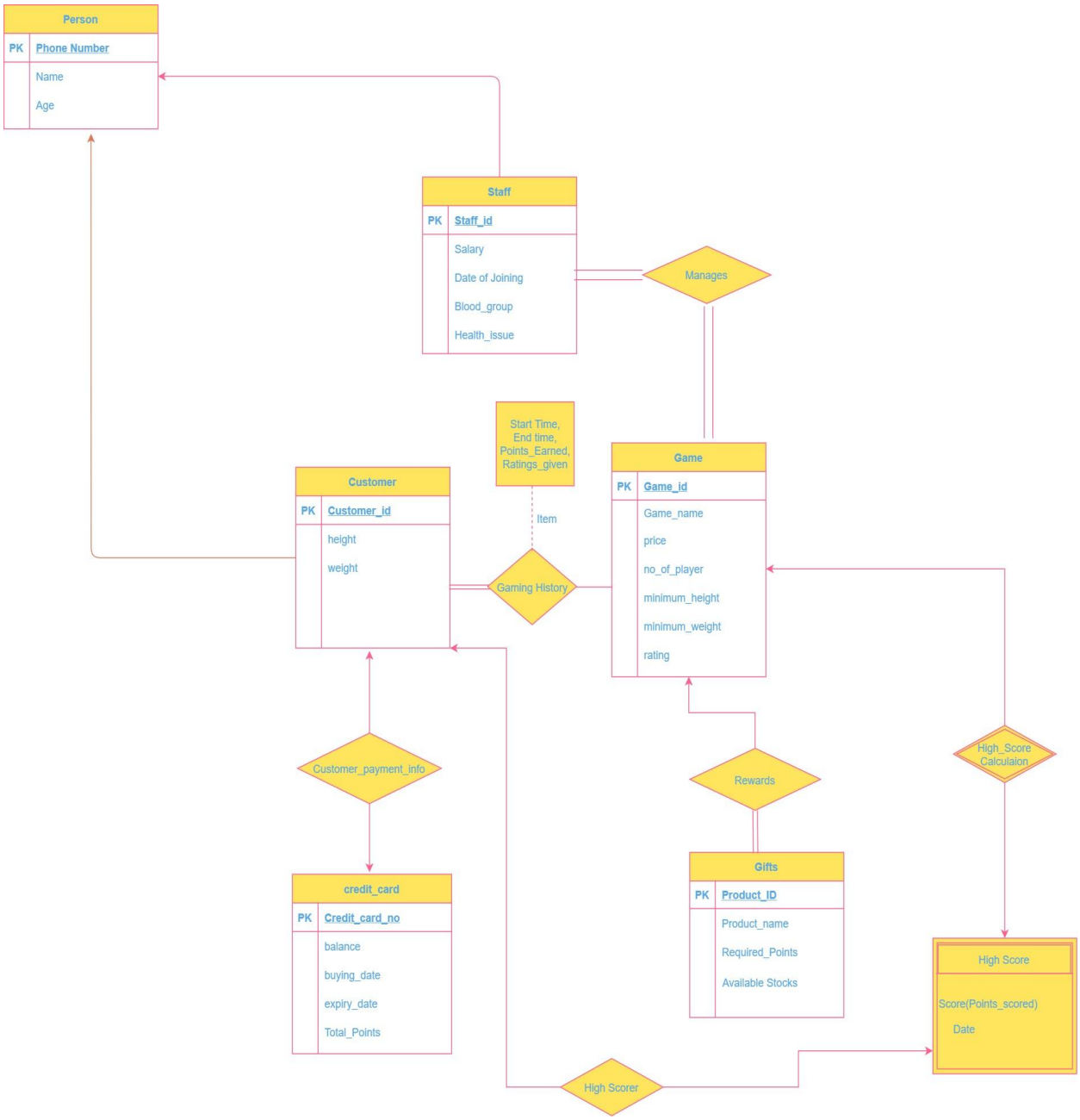
- **Version 1**



- **Version 2:**


Start Time,
End time,
Points_Earned,
Ratings_given
Customer
PK
Customer_id
Customer_name
age
height
weight
phone_number
Gaming History
Game
PK
Game_id
Game_name
price
no_of_player
minimum_height
minimum_weight
rating
Customer_payment_info
Rewards
credit_card
PK
Credit_card_no
balance
buying_date
expiry_date
Total_Points
Gifts
PK
Product_ID
Product_name
required_points
Stock_available

- Final version



Section4: Conversion of Final ER-Diagram to Relational Model.

I. Mapping E-R Model to Relational Model

- Customer(Customer ID, Name, Phone_no, Age, height, weight)
- Game(Game_ID, Game_name, price, Players_required, min_height, min_weight, Rating)
- Staff(Staff_ID, Name, Phone_no, Age, Salary, Joining Date, Blood_group, Health_issue, ManagingGame_ID)
- Game_History(Customer_ID, Game_ID, Start_time, End_time, Points_Earned, Ratings_given)
- Management(Staff_ID, Game_ID)
- Credit_Card(Card_No, Owner_ID, Balance, Buying_Date, Expiry_Date, Total_Points)
- Gifts(Product_ID, Product_Name, Game_ID, Required_points, Available_stocks)
- High_Score(Game_ID, Score, Date, Player_ID)

Section5: Normalization and Schema Refinement.

I.Normalization & Schema Refinement

1. List all the Relations & Schemas with all details

- Customer(Customer ID, Name, Phone_no, Age, height, weight)
- Game(Game_ID, Game_name, price, Players_required, min_height, min_weight, Rating)
- Staff(Staff_ID, Name, Phone_no, Age, Salary, Joining Date, Blood_group, Health_issue)
- Game_History(Customer_ID, Game_ID, Start_time, End_time, Points_Earned, Ratings_given)
- Management(Staff_ID, Game_ID)
- Credit_Card(Card_No, Customer_ID, Balance, Buying_Date, Expiry_Date, Total_Points)
- Gifts(Product_ID, Product_Name, Game_ID, Required_points, Available_stocks)
- High_Score(Game_ID, Score, Date, Customer_ID)

2.Identify and list all types of dependencies (PK, FK, Functional Dependencies) for each relation

- Customer
  - PK - Customer_ID
  - Functional Dependencies
    - Customer_ID ---> Name
    - Customer_ID ---> Phone_No
    - Customer_ID ---> Age
    - Customer_ID ---> Height
    - Customer_ID ---> Weight
- Game
  - PK- Game_ID
  - Functional Dependencies

Game_ID ---> Game_Name
Game_ID ---> Price
Game_ID ---> Players_required
Game_ID ---> min_height
Game_ID ---> min_weight
Game_ID ---> rating
Game_Name ---> Game_ID
Game_Name ---> Price
Game_Name ---> Players_required
Game_Name ---> min_height
Game_Name ---> min_weight
Game_Name ---> rating

- Staff
  - PK- Staff_ID
  - Functional dependencies-
    Staff_ID ---> Name
    Staff_ID ---> phone_no
    Staff_ID ---> age
    Staff_ID ---> salary
    Staff_ID ---> joining_date
    Staff_ID ---> Blood_group
    Staff_ID ---> Health_Issue
    Phone_No ---> Staff_ID

- Game_History
  - PK- Customer_id, Start_time
  - FK- Customer_id, Game_id
  - Functional dependencies-
    Customer_id, Start_time ---> End_time
    Customer_id, Start_time ---> Game_id
    Customer_id, Start_time---> Points_Earned
    Customer_id, Start_time ---> Ratings_Given

- Management
  - PK- Staff_ID,Game_ID
  - FK - Staff_ID, Game_ID
  - Functional Dependencies
    Staff_ID,Game_ID ---> Staff_ID
    Staff_ID,Game_ID ---> Game_ID

- Credit card
  - PK- Card_no
  - FK- Customer_ID
  - Functional Dependencies
    Card_ No ---> Customer_ID
    Card_ No ---> Balance
    Card_ No ---> Buying_Date

Card_ No ---> Expiry_Date
Card_ No ---> min_weight
Card_ No ---> total_points
Customer_ID ---> Balance
Customer_ID---> Buying_Date
Customer_ID---> Expiry_Date
Customer_ID---> min_weight
Customer_ID---> total_points
Customer_ID---> Card_No

- Gifts
    - PK- Product_id
    - FK- Game_id
    - Functional dependencies-
        Product_id---> Required_Points
        Product_id ---> Available Stocks
        Product_id ---> Product_Name
        Product_name ---> Required_Points
        Product_Name ---> Available Stocks
        Product_Name--->Product_id

- High_Score
    - PK- Game_id
    - FK- Game_id,Player_ID
    - Functional dependencies-
        Game_id ---> Score
        Game_id ---> Date
        Game_id ---> customer_id
        customer_id ---> Score
        customer_id ---> Date
        customer_id---> game_id

3) Investigate every schema for the following List of redundancies existing for every schema which is part of the database (document it). List of update, delete, and insert anomalies for every schema (document it)

- Customer
    - Insert anomalies - As we can insert data into the customer table the same information should also be inserted in the credit_card table so this leads to insert anomalies in the customer table.
    - Delete anomalies - Customer_Id is referenced by game_history, credit_card and high_score table. So there will be delete anomalies when we want to delete some customer information and the databaser may be inconsistent .
    - Update anomalies - Customer_Id is referenced by game_history, credit_card and high_score table. So there will be update anomalies when we want to update some customer information and the database may be inconsistent .

- Game
    - Insert anomalies - There are no insert anomalies as we can insert data into the game table without any problem.
    - Delete anomalies - Game_Id is referenced by game_history, management, gfit and high_score table. So there will be delete anomalies when we want to delete some game information and the database may be inconsistent .
    - Update anomalies - Game_Id is referenced by game_history, management, gfit and high_score table. So there will be update anomalies when we want to update some game information and the database may be inconsistent .

- Staff
    - Insert anomalies - Staff_id is referenced by management. So there will be insert anomalies when we want to insert new staff members and the database will be inconsistent.
    - Delete anomalies - Staff_id is referenced by management. So there will be delete anomalies when we want to delete any staff members and the database will be inconsistent.
    - Update anomalies - There will be no anomalies in the staff table when we update the information of staff members as only staff_id (primary key) is referenced by management.

- Game_History
    - Insert Anomalies: Insertion of a game history tuple includes the game played, customer, points_earned and rating given by the customer at the end. The balance in the customer's credit card should be deducted according to the price of the game played.
      Ratings of the game must be updated and points earned by the customer are to be added to the total points in the customer's card.
      Also, changes in the high score table must be done if the points earned are the highest till date.
    - Delete Anomalies: No delete anomaly will be there as a game history will never be deleted.
    - Update Anomalies: No update anomaly will be there as a game history will never be updated.

- Management
    - Insert Anomalies - When information is inserted in the management table, staff_id and game_id should be there in staff and game table. So insert anomalies will arise.
    - Delete Anomalies - Staff_id and Game_id are referenced from staff and game tables. So when information is deleted from the management table, delete anomalies will arise and staff, game tables will be in an inconsistent state.
    - Update Anomalies - Staff_id and Game_id are referenced from staff and game table. So when information is updated in the management table, this information should also be updated in the corresponding table. Hence update anomalies will be there.

- Credit_card
    - Insert Anomalies - When information is inserted in the credit_card table, customer_id should be there in the customer table. So insert anomalies will arise.
    - Delete Anomalies - customer_id is referenced from customer table. So when information is deleted from the credit_card table, delete anomalies will arise and the customer table will be in an inconsistent state.
    - Update Anomalies - No update anomalies will be there.
- Gifts
    - Insert anomalies - when we insert some new gifts in table the games which associated with this new gift must be in the game table so insert anomalies arise.
    - Delete anomalies - No delete anomaly will be there as we delete some gifts there will be no problem in another table .
    - Update anomalies - No update anomaly will be there as we delete some gifts there will be no problem in another table .

- High_score
    - Insert anomalies - when some game's high score is inserted, the corresponding game_id should be there in the game table. So insert anomalies will arise.
    - Delete anomalies - no delete anomalies.
    - Update anomalies - no update anomalies.

4) Normalize the database up to 1NF (scalar values)

- Customer
    - Name is a composite attribute. So not in 1NF.
    - Customer(Customer ID, First_Name,Middle_Name, Last_Name, Phone_no, Age, height, weight)
- Game
    - Already in 1NF

- Staff
    - Name is a composite attribute. So not in 1NF.
    - Staff(Staff_ID, First_Name,Middle_Name, Last_Name, Phone_no, Age, Salary, Joining Date, Blood_group, Health_issue)
- Game_History
    - All the attributes are atomic.Hence,in 1NF

- Management
    - All the attributes are atomic.Hence,in 1NF

- Credit_Card
    - All the attributes are atomic.Hence,in 1NF
- Gifts
    - All the attributes are atomic.Hence,in 1NF

- High_Score
    - All the attributes are atomic.Hence,in 1NF

5) Normalize the database further to 2NF (Remove Partial Dependencies)

- Customer
    - There is only one candidate key, which is Customer_ID. Customer_ID is determining all other non-prime attributes.
    - No proper subset of candidate key is determining non prime attributes.
    - So the customer table is already in 2NF.

- Game
    - Candidate keys are Game_ID and Game_Name. There are no proper subsets for these keys to leave scope for any partial dependencies, so the relation is in 2NF.

- Game_History
    - Candidate keys can be (Customer_id, Start_time), (Customer_id, End_time).
    - It can be proved that any proper subset of any of the candidate keys doesn't determine any non-prime attribute.
    - So, the relation is in 2NF.

- Management
    - Candidate key is (staff_id,game_id)
    - There is no prime attribute
    - Here any proper subset of candidate key doesn't determine any non prime attribute.
    - So the relation is in 2NF.

- Credit_Card
    - Candidate keys are card_no, owner_id
    - Here no proper subset of candidate key is determining non prime attributes.
    - So the relation is in 2NF.

- Gifts
    - Candidate keys are product_id, product_name
    - Here no proper subset of candidate key is determining non prime attributes.
    - So the relation is in 2NF.

- High_score
    - Candidate keys are game_id, customer_id
    - Here no proper subset of candidate key is determining non prime attributes.
    - So the relation is in 2NF

6) Identify (and document) List of redundancies exiting for the schema in 2NF

- There are no redundancies in schema in 2NF as mentioned above.

7) Normalize it further to 3NF/BCNF (Remove Transitive Dependencies)

- In all the schemas, there are no transitive dependencies, which means no non-prime attribute determines any non-prime attribute.
- So all the schemas are in 3NF.
- In all the schemas, determinants in all functional dependencies are candidate keys.

Section6: SQL: Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of output of each query.

1. Final DDL Scripts.

1) Recreate database by writing all Create Table statements (DDL) to accommodate the new design which is in 3NF/BCNF (removing your original version of relations)

- **Customer**

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."Customer"
(
    "Customer_ID" bigint NOT NULL,
    "First_Name" character varying NOT NULL,
   "Middle_Name" character varying NOT NULL,
   "Last_Name" character varying NOT NULL,
    "Phone_no" character varying NOT NULL,
    "Age" integer NOT NULL,
    "Height" integer NOT NULL,
    "Weight" integer NOT NULL,
    PRIMARY KEY ("Customer_ID")

  );
```

- **Game**

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."Game"
(
   "Game_ID" bigint NOT NULL,
    "Game_Name" character varying NOT NULL,
    "Price" money,
    "Players_Required" integer,
    "Min_Height" integer,
    "Min_Weight" integer,
    "Rating" integer,
    PRIMARY KEY ("Game_ID")
);
```

- **Staff**

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."Staff"
(
    "Staff_ID" bigint NOT NULL,
    "First_Name" character varying NOT NULL,
    "Middle_Name" character varying NOT NULL,
    "Last_Name" character varying NOT NULL,
    "Phone_No" character varying NOT NULL,
    "Age" integer,
    "Salary" integer,
    "Joining_Date" date,
    "Blood_Group" character varying,
    "Health_Issue" character varying,
    CONSTRAINT "Staff_Pkey" PRIMARY KEY ("Staff_ID")
);
```

Game History Script

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."Game_History"
(
    "Customer_ID" bigint NOT NULL,
    "Game_ID" bigint NOT NULL,
    "Start_time" timestamp without time zone NOT NULL,
    "End_time" timestamp without time zone NOT NULL,
    "Points_earned" integer,
    "Ratings_given" integer,
    CONSTRAINT "Game_History_pkey" PRIMARY KEY ("Customer_ID", "Start_time"),
    CONSTRAINT "Customer_fkey" FOREIGN KEY (“Customer_ID")
    REFERENCES "Game_Zone"."Customer" ("Customer_ID") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE SET NULL,
    CONSTRAINT "game_idkey" FOREIGN KEY ("Game_ID")
    REFERENCES "Game_Zone"."Game" ("Game_ID")
    ON UPDATE NO ACTION
    ON DELETE SET NULL
)
```

- **Management script**

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."Management"
(
    "Staff_ID" bigint NOT NULL,
    "Game_ID" bigint NOT NULL,
    CONSTRAINT "Manage_pkey" PRIMARY KEY ("Staff_ID", "Game_ID"),
    CONSTRAINT "Staff_fkey" FOREIGN KEY ("Staff_ID")
        REFERENCES "Game_Zone"."Staff" ("Staff_ID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "Game_fkey" FOREIGN KEY ("Game_ID")
```

```
        REFERENCES "Game_Zone"."Game" ("Game_ID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE
        NOT VALID
);
```

- **Credit Card Script**

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."Credit_Card"
(
    "Card_No" bigint,
    "Customer_ID" bigint,
    "Balance" money,
    "Buying_date" date,
    "Expiry_date" date,
    "Total_Points" integer,
    CONSTRAINT "Card_pkey" PRIMARY KEY ("Card_No"),
    CONSTRAINT "Owner_fkey" FOREIGN KEY ("Customer_ID")
        REFERENCES "Game_Zone"."Customer" ("Customer_ID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE
        NOT VALID
);
```

- **Gifts Script**

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."Gifts"
(
    "Product_ID" bigint NOT NULL,
    "Product_name" character varying,
    "Game_ID" bigint,
    "Required_points" integer,
    "Available_Stock" integer,
    CONSTRAINT "Gift_pkey" PRIMARY KEY ("Product_ID"),
    CONSTRAINT "Game_fkey" FOREIGN KEY ("Game_ID")
        REFERENCES "Game_Zone"."Game" ("Game_ID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE SET NULL
        NOT VALID
);
```

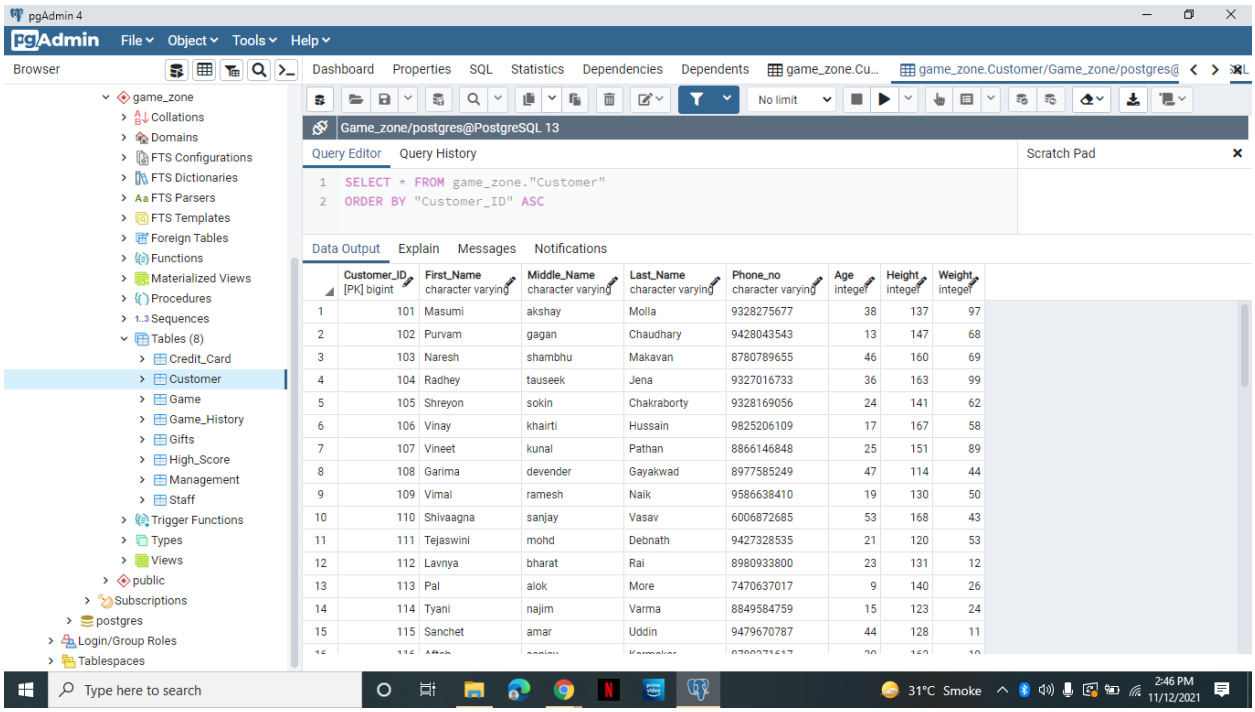
- **High score Script**

```
CREATE TABLE IF NOT EXISTS "Game_Zone"."High_Score"
(
    "Game_ID" bigint NOT NULL,
    "Highest_score" integer,
    "Date" date,
```

```
    "Customer_ID" bigint,
    CONSTRAINT high_socre_fkey PRIMARY KEY ("Game_ID"),
    CONSTRAINT game_fkey FOREIGN KEY ("Game_ID")
        REFERENCES "Game_Zone"."Game" ("Game_ID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT player_fkey FOREIGN KEY ("Customer_ID")
        REFERENCES "Game_Zone"."Customer" ("Customer_ID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE SET NULL
        NOT VALID
);
```

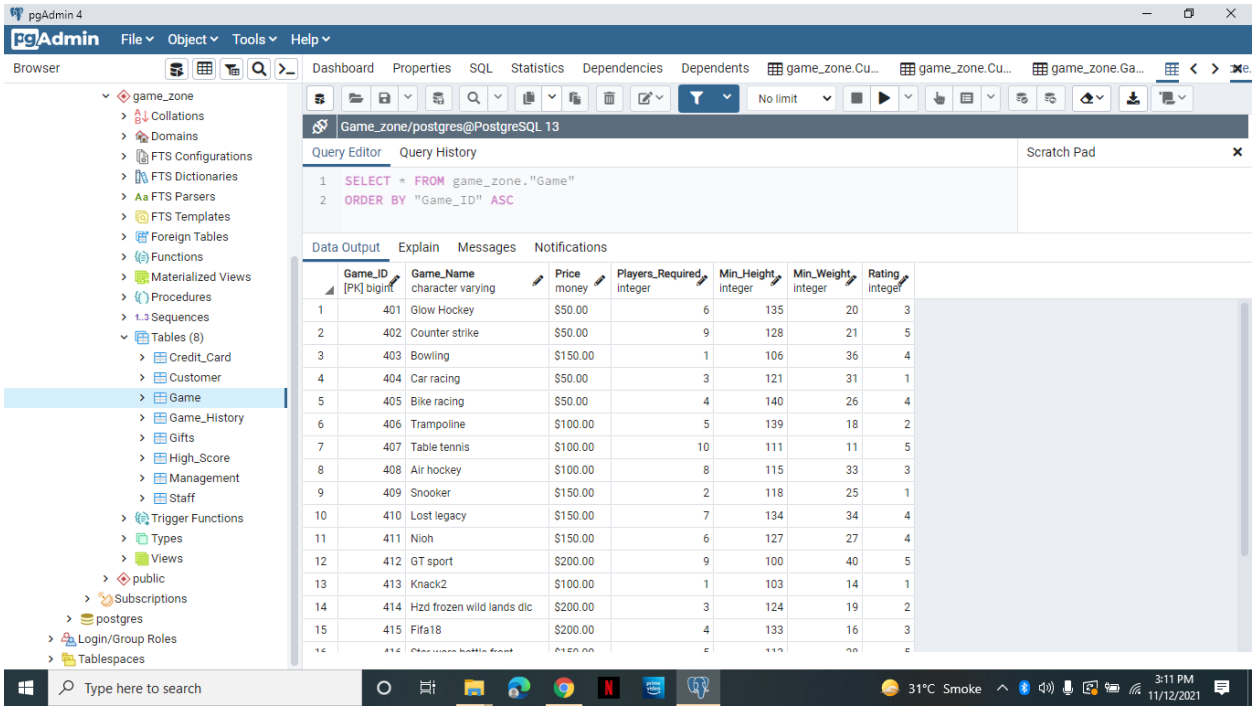
3) Create instances of this new database by populating it using appropriate INSERT INTO statements /using scripts. Make sure that every table has at least 80-100 tuples.

- Customer table



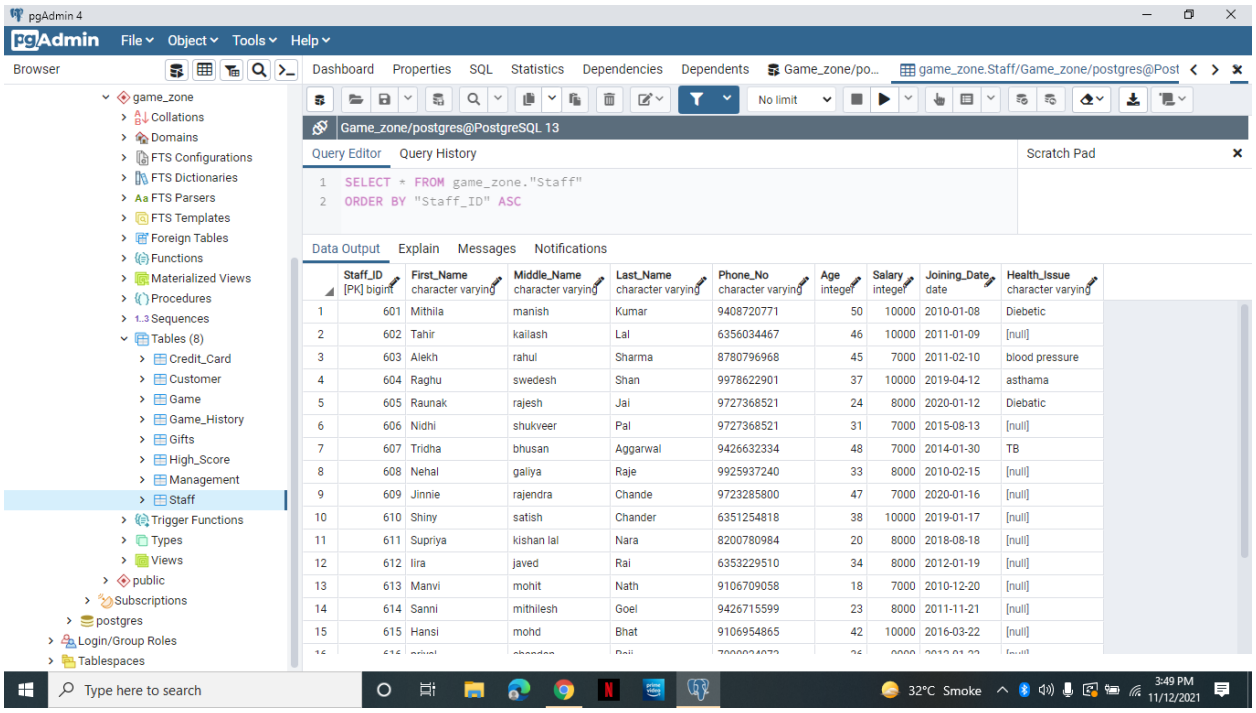
No of Tuples = 60

- Game table



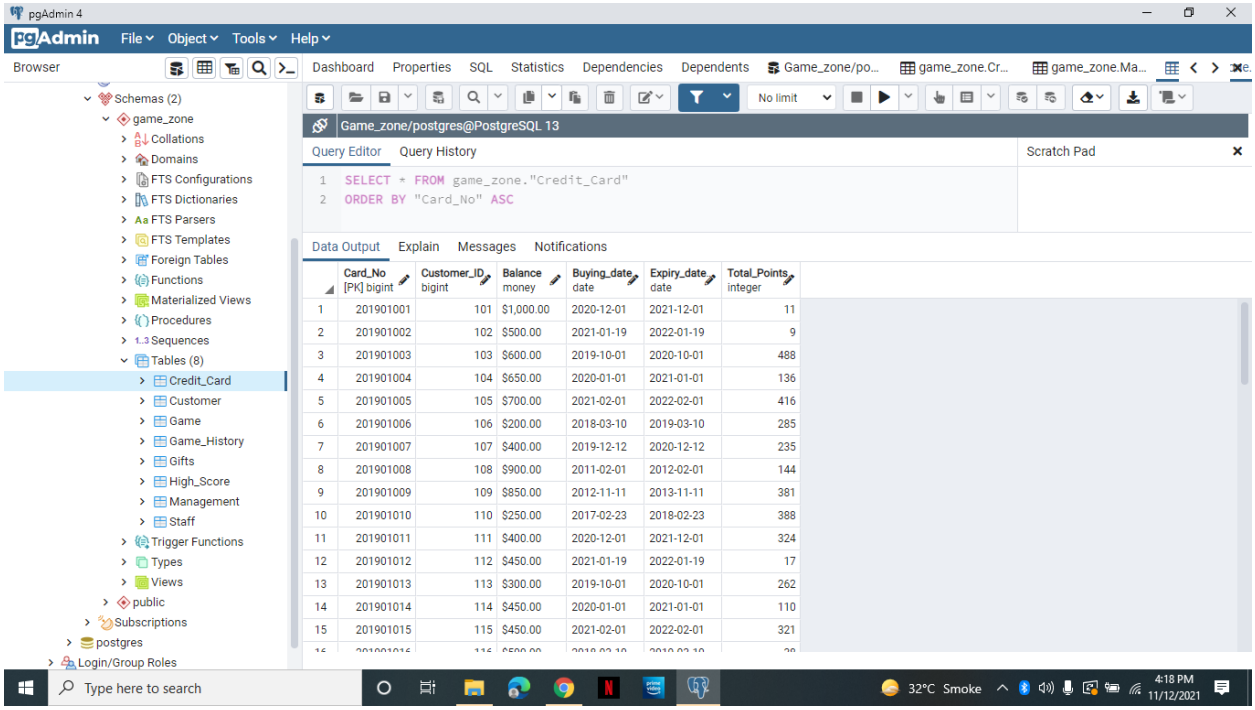
No of tuples = 20

Staff table



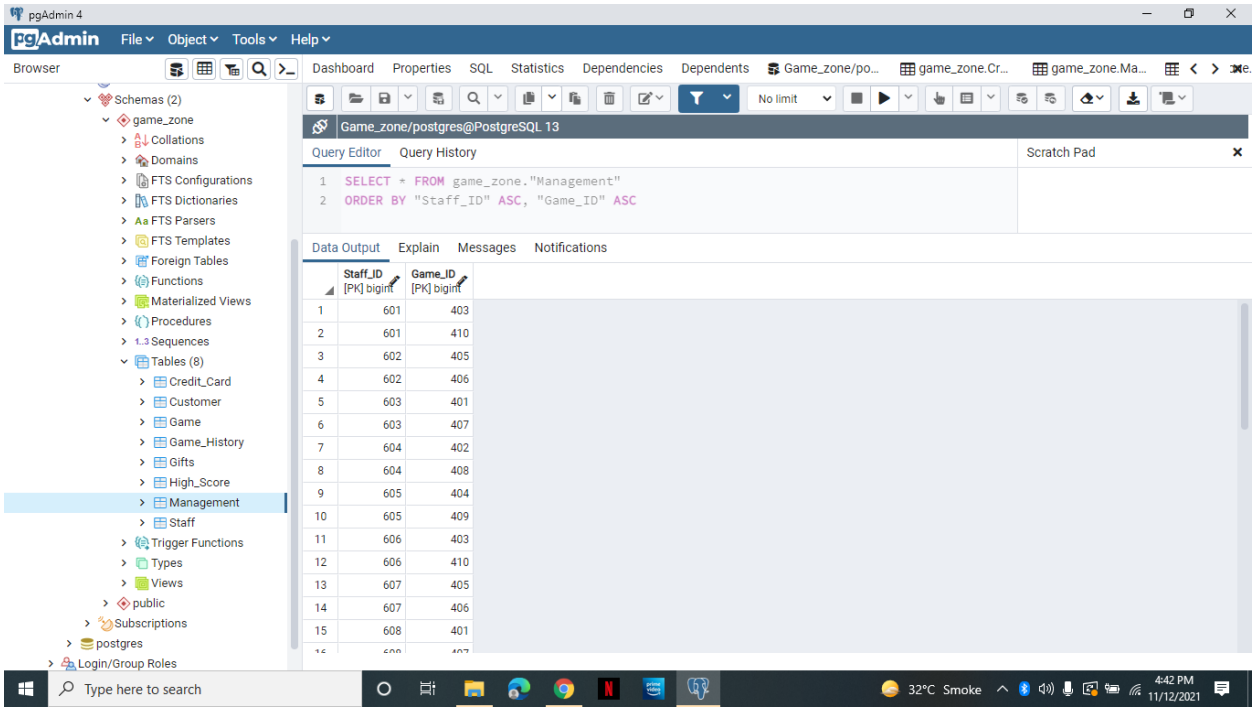
No of tuples = 20

- Credit Card table



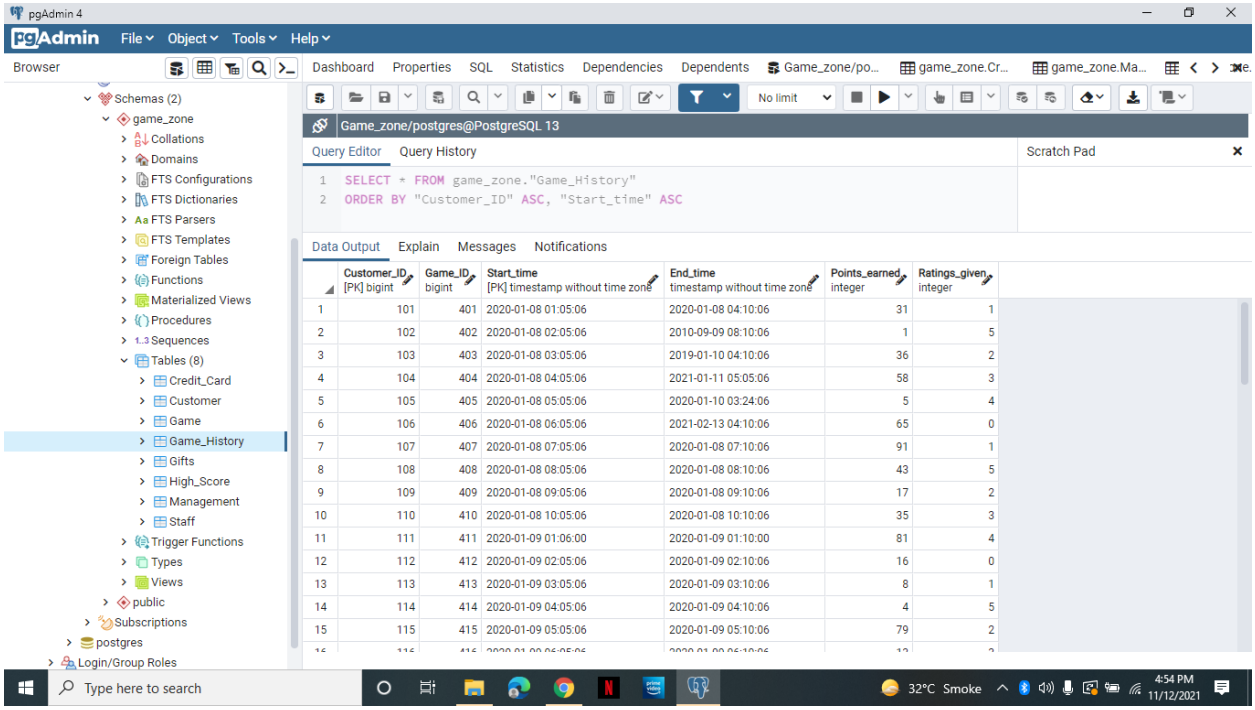
No of tuples = 60

- Management table



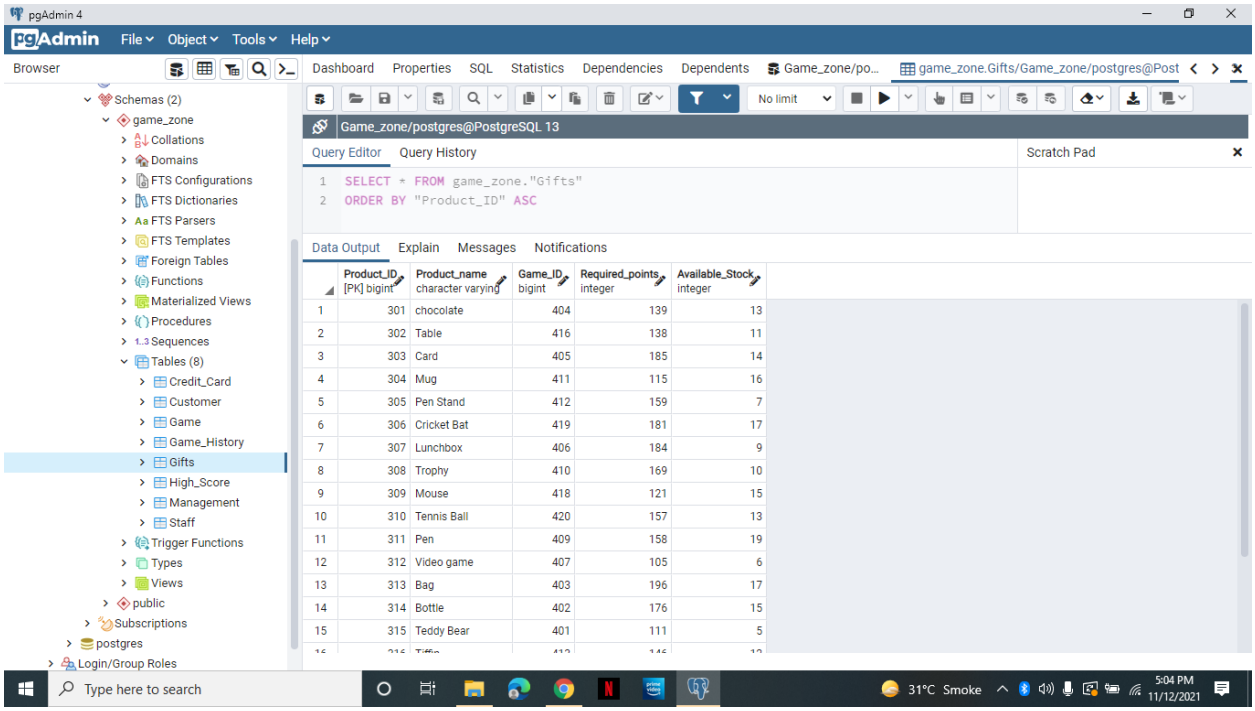
No of tuple = 40

- Game history table



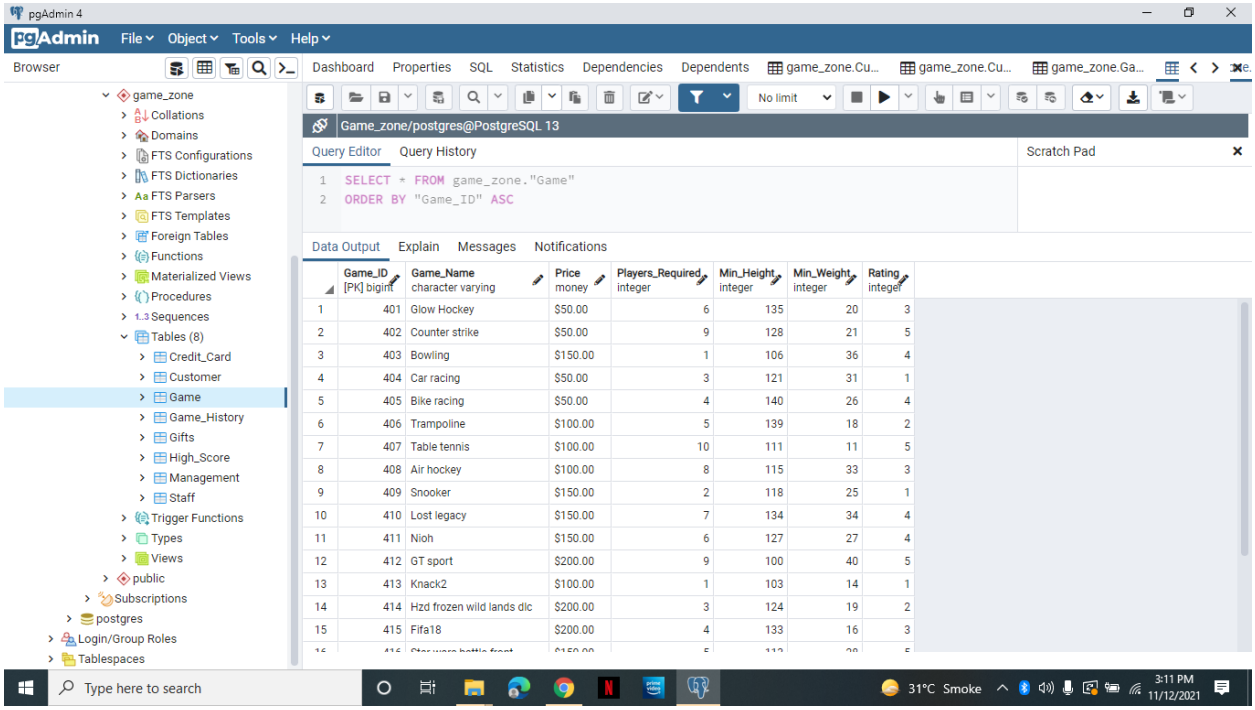
No of tuples = 60

- Gift table



No of tuples = 20

- High score table



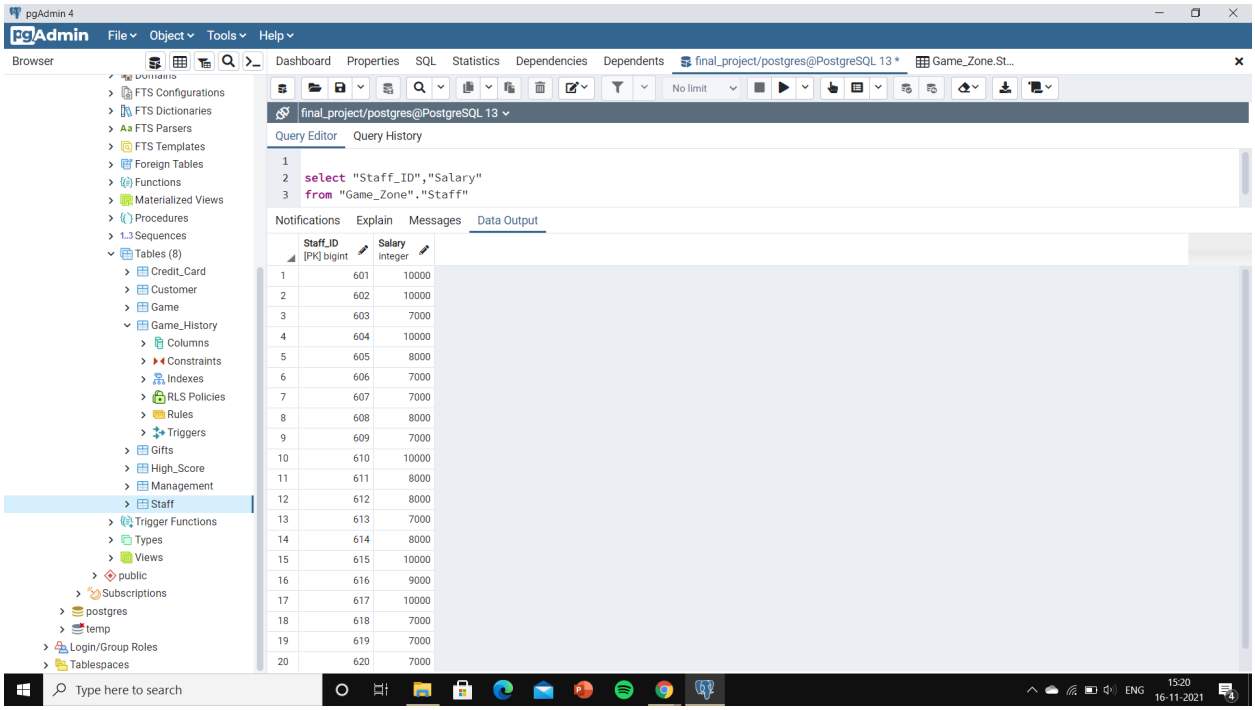
Game_ID [PK] bigint	Game_Name character varying	Price money	Players_Required integer	Min_Height integer	Min_Weight integer	Rating integer
401	Glow Hockey	$50.00	6	135	20	3
402	Counter strike	$50.00	9	128	21	5
403	Bowling	$150.00	1	106	36	4
404	Car racing	$50.00	3	121	31	1
405	Bike racing	$50.00	4	140	26	4
406	Trampoline	$100.00	5	139	18	2
407	Table tennis	$100.00	10	111	11	5
408	Air hockey	$100.00	8	115	33	3
409	Snooker	$150.00	2	118	25	1
410	Lost legacy	$150.00	7	134	34	4
411	Nioh	$150.00	6	127	27	4
412	GT sport	$200.00	9	100	40	5
413	Knack2	$100.00	1	103	14	1
414	Hzd frozen wild lands dlc	$200.00	3	124	19	2
415	Fifa18	$200.00	4	133	16	3

No of tuples = 20

- **SQL Queries**

1. Find staff id and it's salary.
   SQL query :

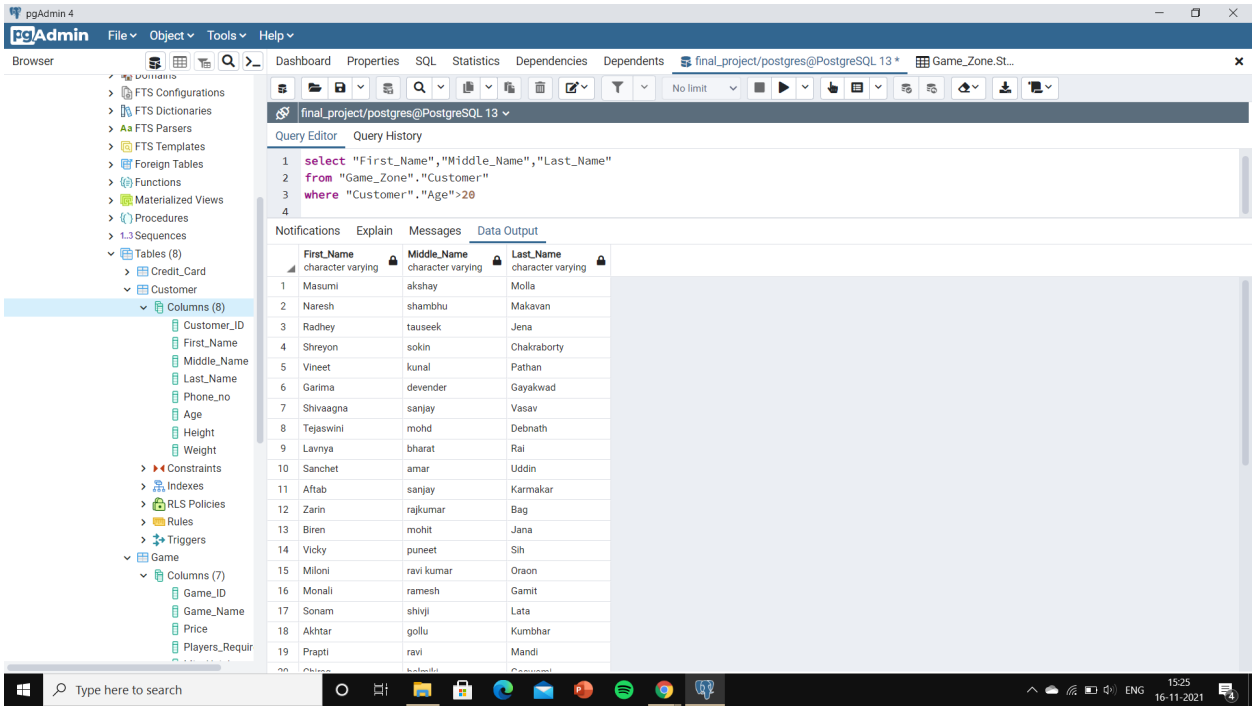
```
select "Staff_ID","Salary"
from "Game_Zone"."Staff"
```



No of Tuples = 20

2. Find a customer's name whose age is greater than 20.
SQL query:

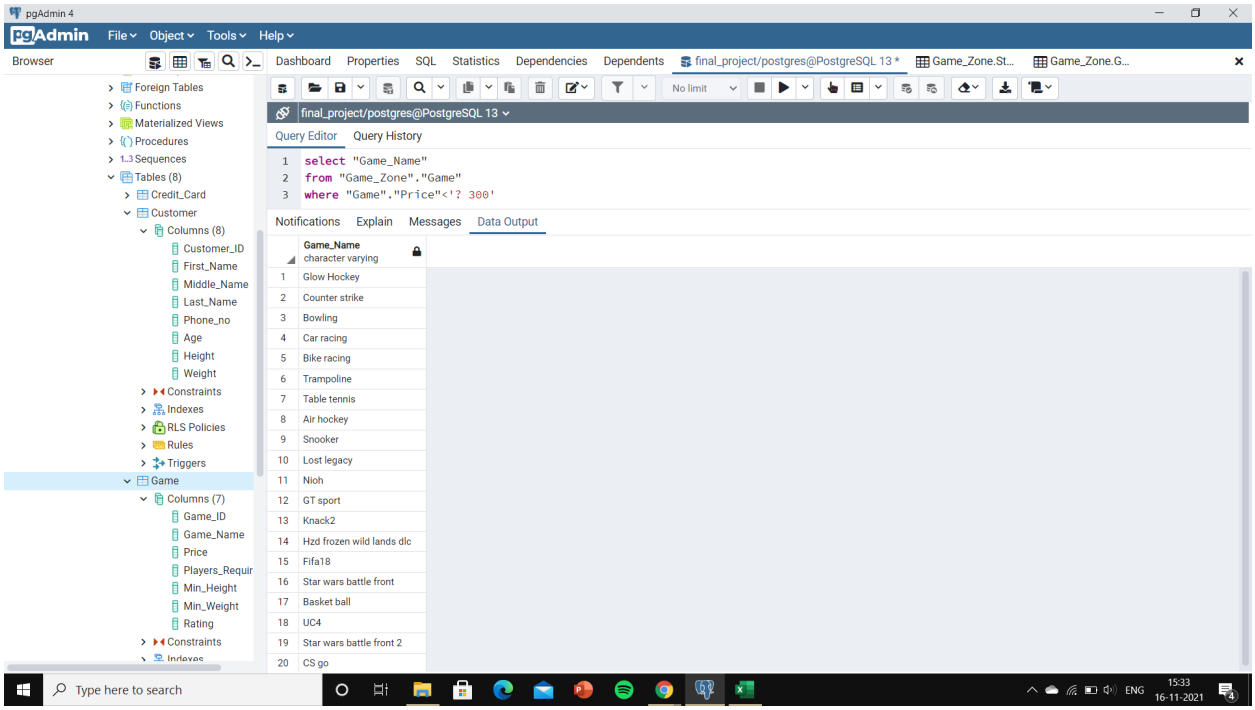
```
select "First_Name","Middle_Name","Last_Name"
            from "Game_Zone"."Customer"
            where "Customer"."Age">20
```



No of Tupels = 40

3. Find the name of games whose price is less than 300.
   SQL query:

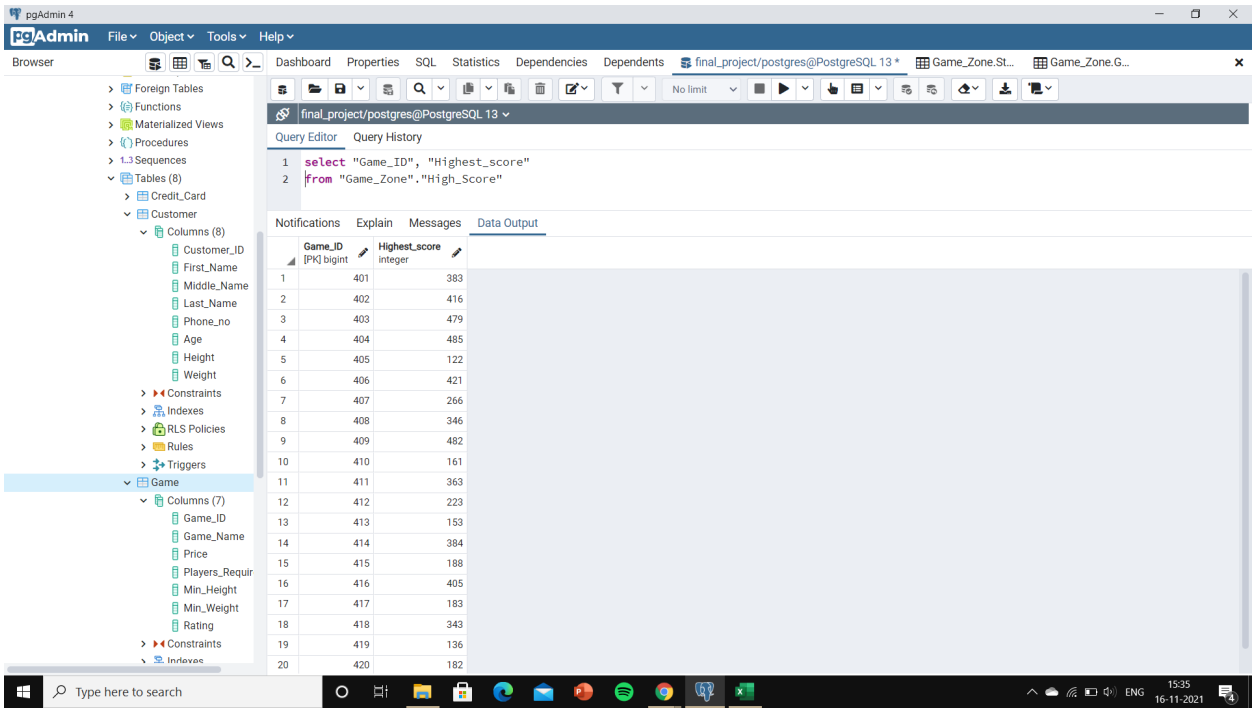
```
select "Game_Name"
from "Game_Zone"."Game"
where "Game"."Price"<'$ 300'
```



No. of Tuples = 20

4. Find a high score for each game.
SQL query :

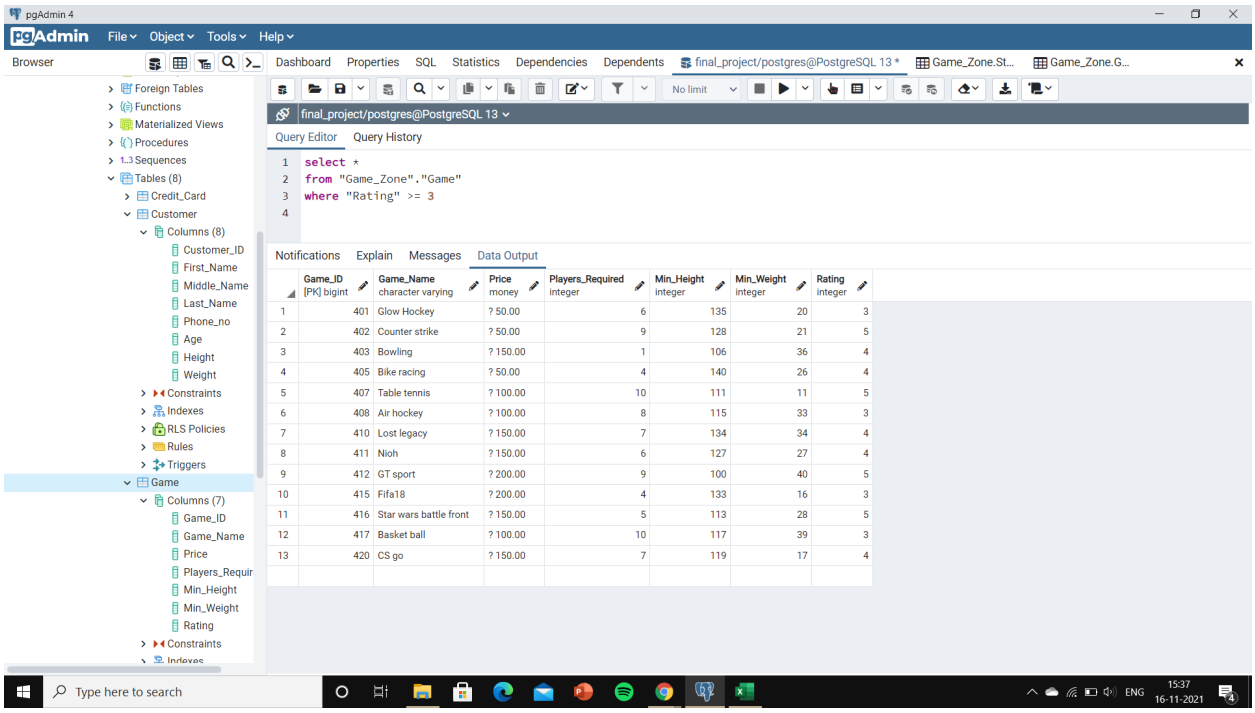
```
select "Game_ID", "Highest_score" from "Game_Zone"."High_Score"
```



No. of Tuples = 20

5. Find the games whose rating is at least 3.
SQL Query :

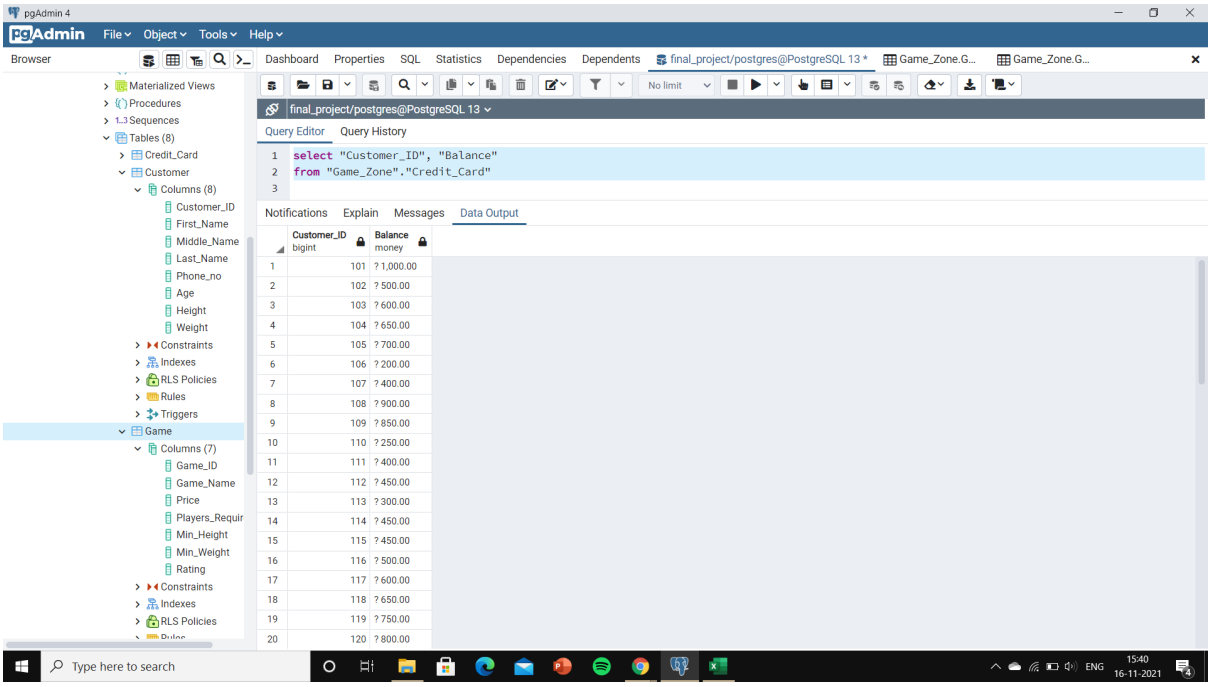
```
select *
from "Game_Zone"."Game"
where "Rating" >= 3
```



No. of Tuples = 13

6. Find the balance of each customer's credit card.
SQL query :

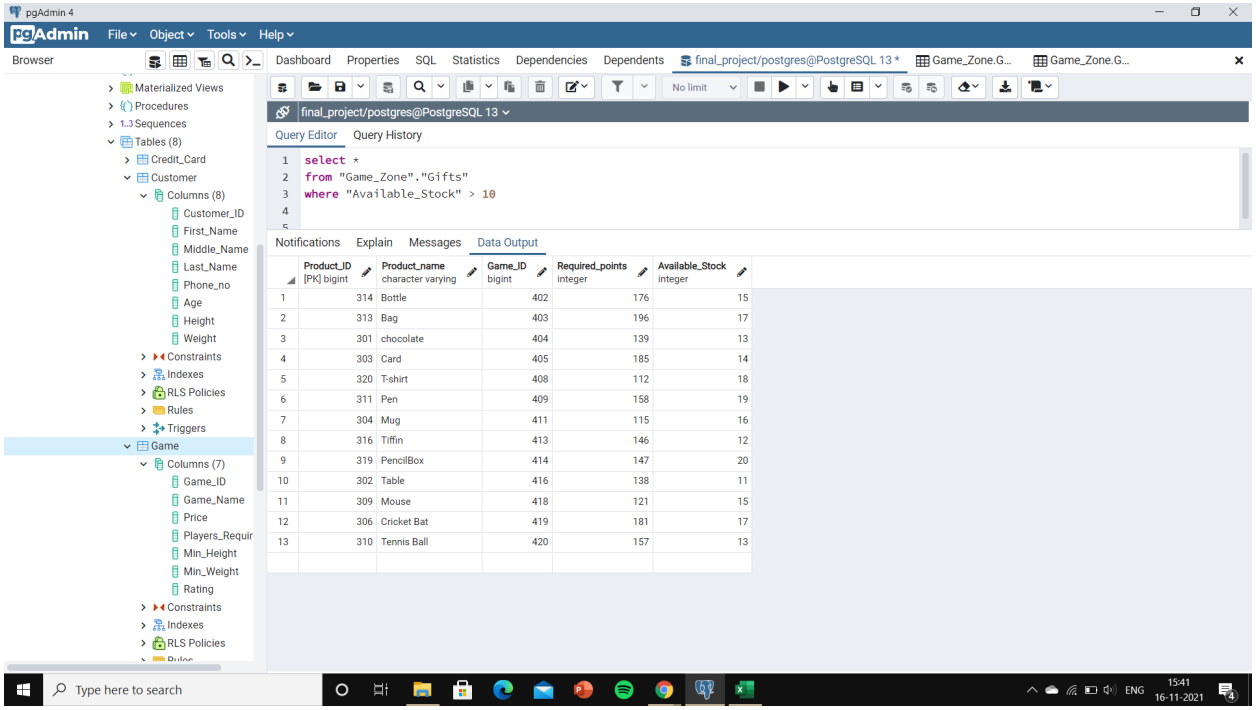
```
select "Customer_ID", "Balance"
from "Game_Zone"."Credit_Card"
```



No of Tuples = 60

7. Find the gifts whose available stock is greater than 10.
   SQL query :

```
select *
from "Game_Zone"."Gifts"
where "Available_Stock" > 10
```

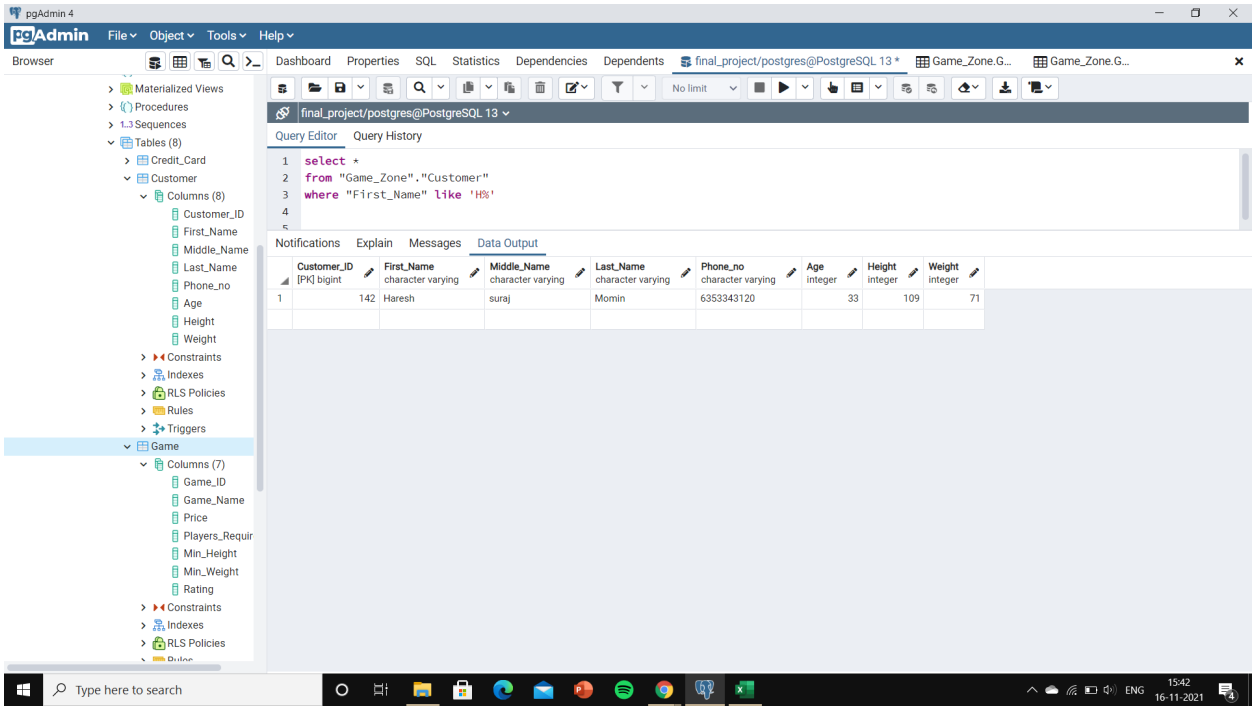


	Product_ID [PK] bigint	Product_name character varying	Game_ID bigint	Required_points integer	Available_Stock integer
1	314	Bottle	402	176	15
2	313	Bag	403	196	17
3	301	chocolate	404	139	13
4	303	Card	405	185	14
5	320	T-shirt	408	112	18
6	311	Pen	409	158	19
7	304	Mug	411	115	16
8	316	Tiffin	413	146	12
9	319	PencilBox	414	147	20
10	302	Table	416	138	11
11	309	Mouse	418	121	15
12	306	Cricket Bat	419	181	17
13	310	Tennis Ball	420	157	13

No. of Tuples = 13

8. Find the customers whose first name starts with ‘H’.
   SQL query :

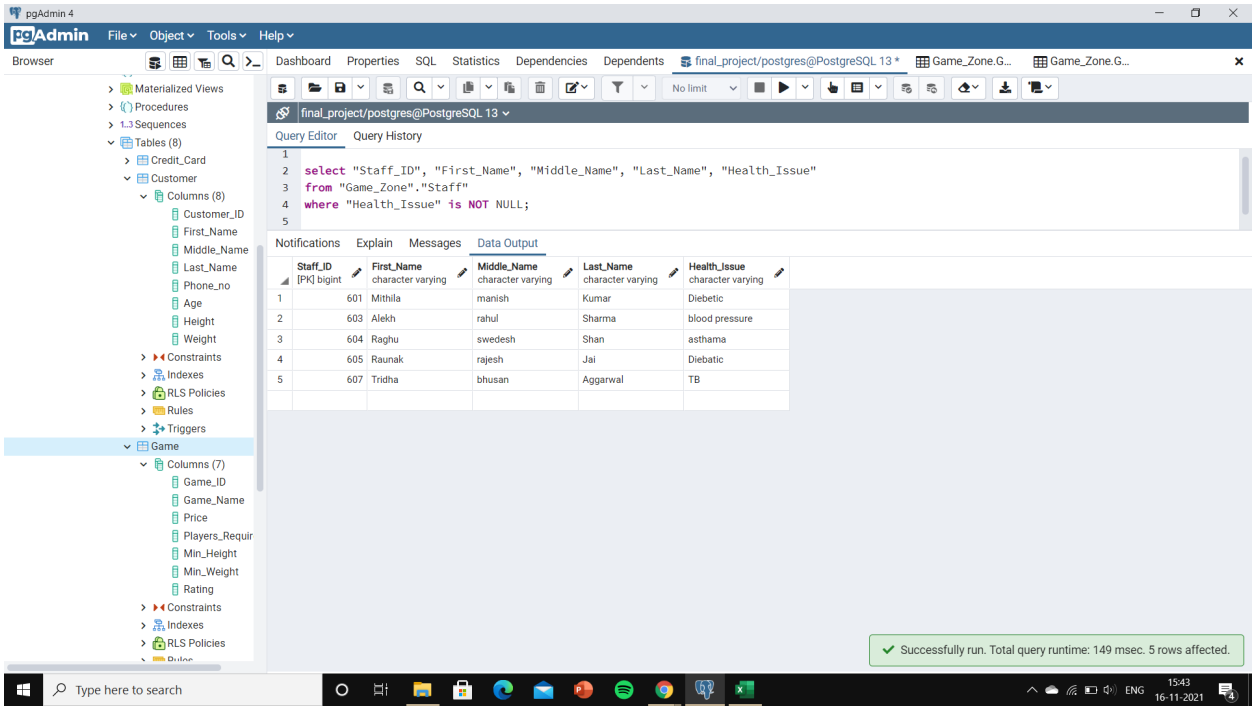
```
select *
from "Game_Zone"."Customer"
where "First_Name" like 'H%'
```



No. of Tuples = 1

9. Find the staff_id, name, health_issue of the staff members who have a health issue.
   SQL query :

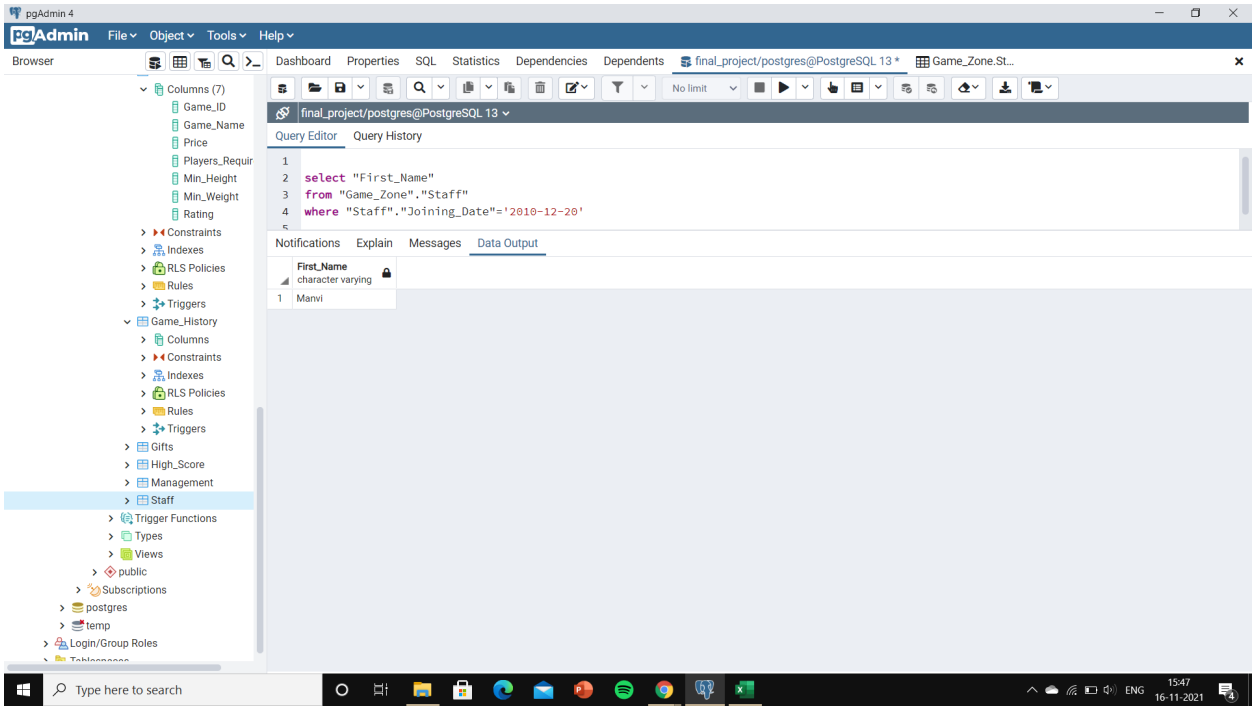
```
select "Staff_ID", "First_Name", "Middle_Name", "Last_Name",
"Health_Issue"
from "Game_Zone"."Staff"
where "Health_Issue" is NOT NULL;
```



No. of Tuples = 5

10. Find the name of the staff member who joined at 2010-12-20.
    SQL query :

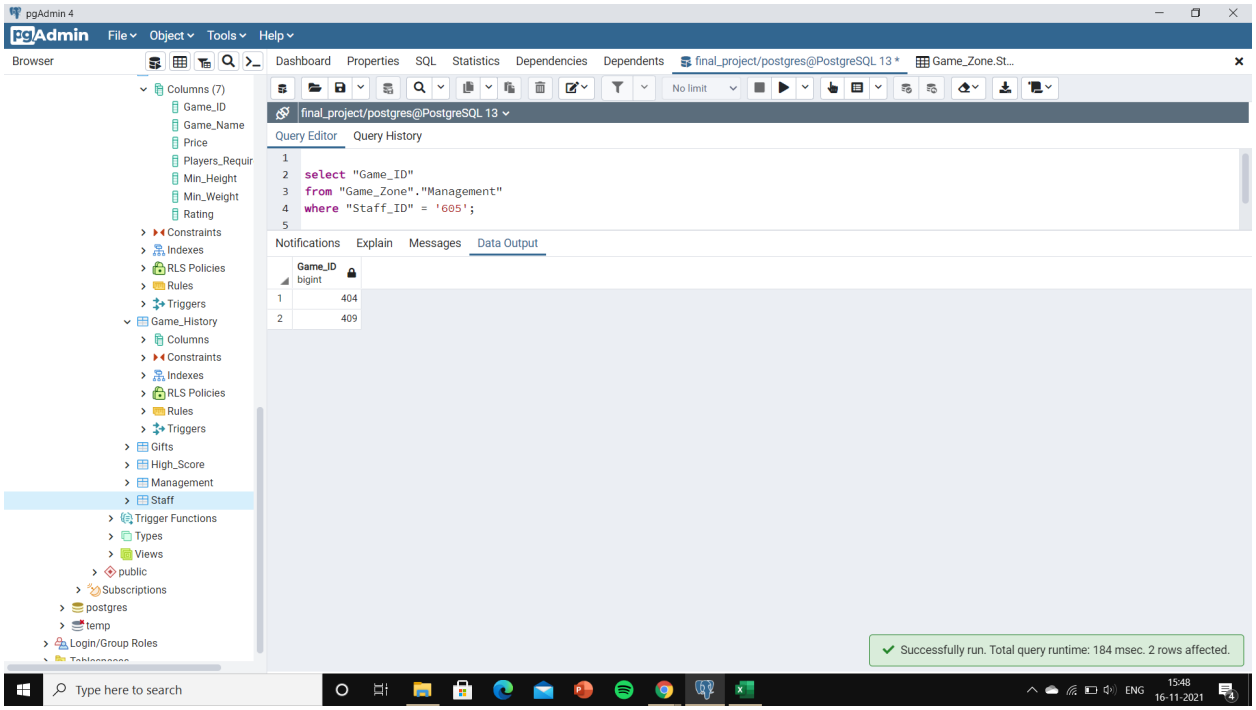
```
select "First_Name","Middle_Name","Last_Name"
from "Game_Zone"."Staff"
where "Staff"."Joining_Date"='2010-12-20'
```



NO of Tuples =1

11. Find the games id which is associated with staff id 615.
SQL query :

```
select "Game_ID"
from "Game_Zone"."Management"
where "Staff_ID" = '605';
```

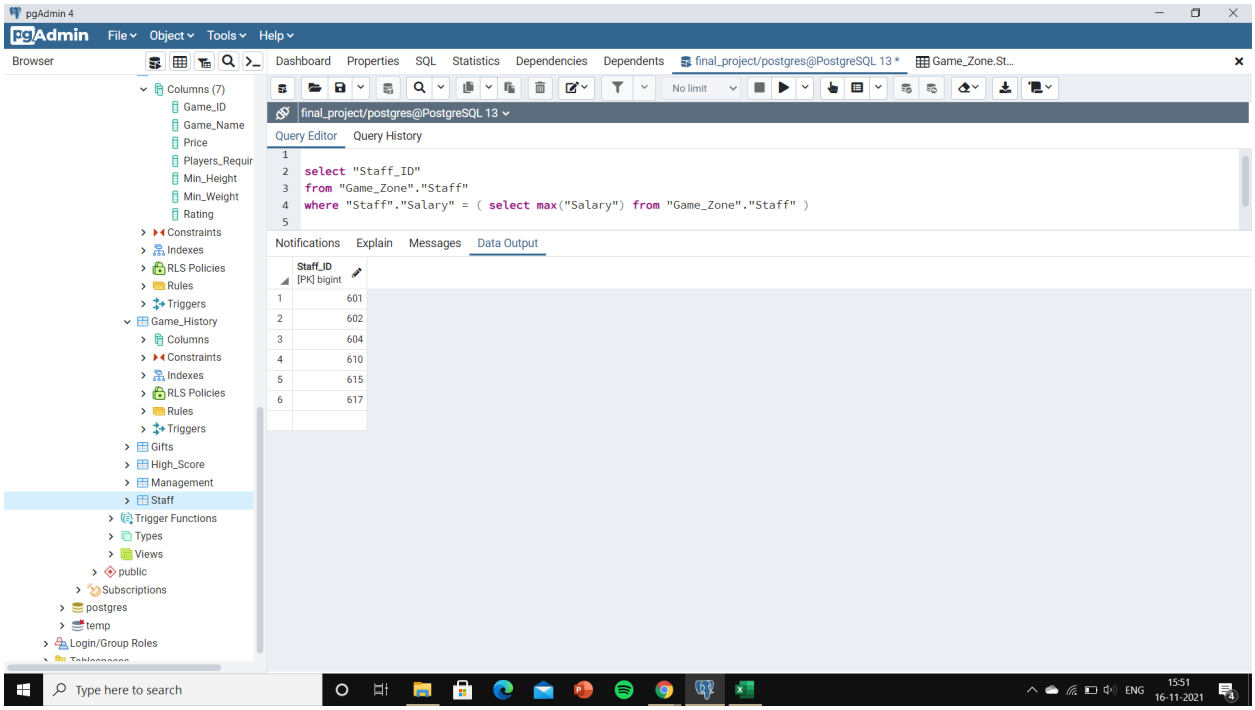


No. Of Tuples = 2

12. Find all the staff_id who has the highest salary.

SQL query:

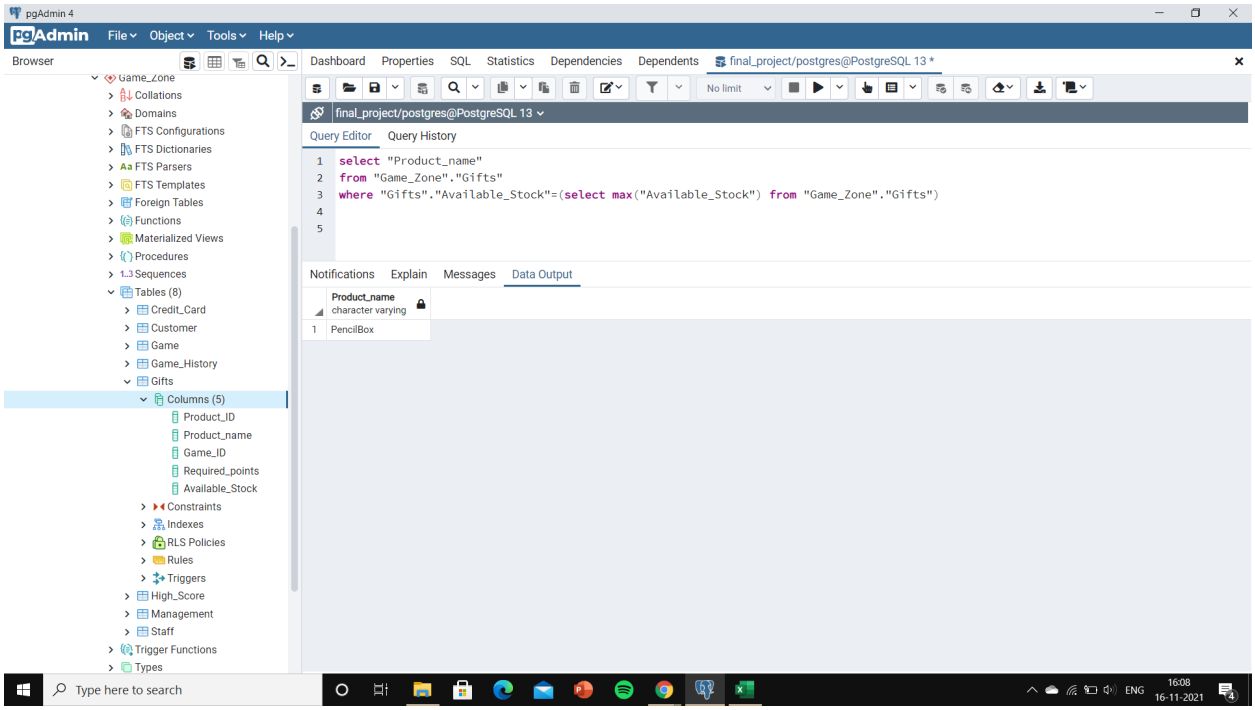
```
select "Staff_ID"
from "Game_Zone"."Staff"
where "Staff"."Salary" = ( select max("Salary") from "Game_Zone"."Staff"
)
```



No. of Tuples = 6

13. Find the gifts name which available in highest amount.
SQL query:

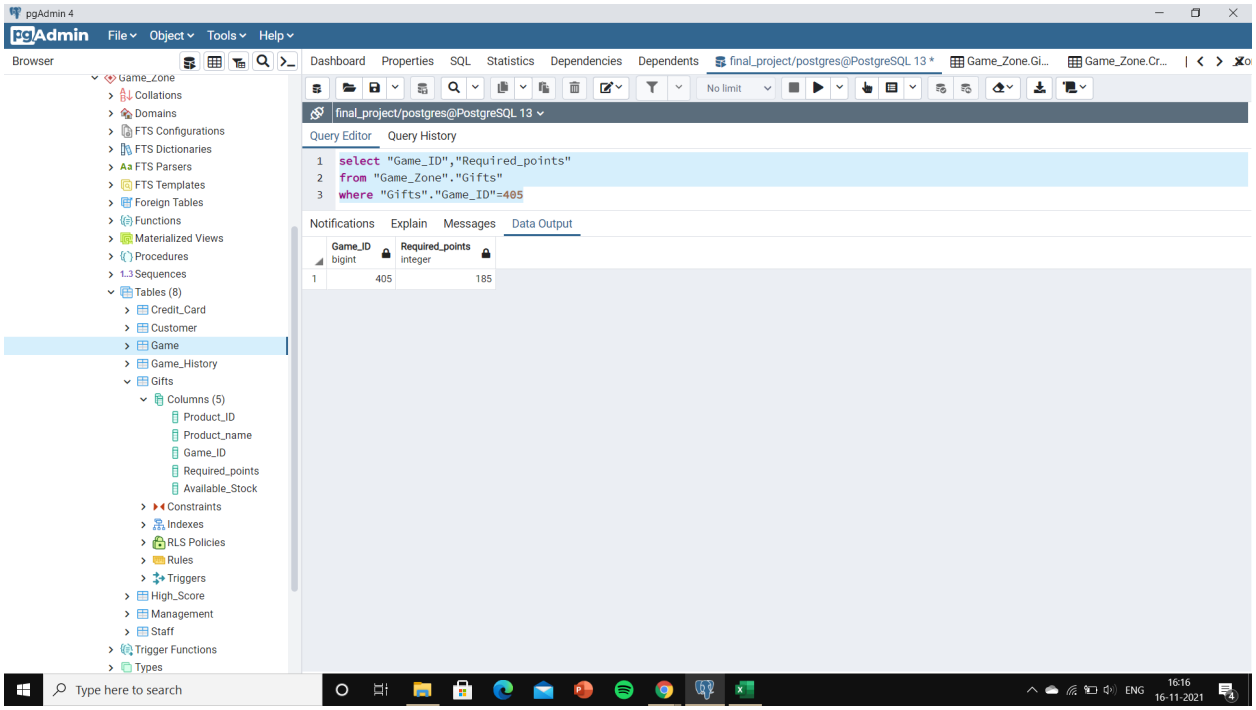
```
select "Product_name"
from "Game_Zone"."Gifts"
where "Gifts"."Available_Stock"=(select max("Available_Stock") from
"Game_Zone"."Gifts")
```



No. of tuples =1

14. Find the required points to win a gift for game id 405.
SQL query:

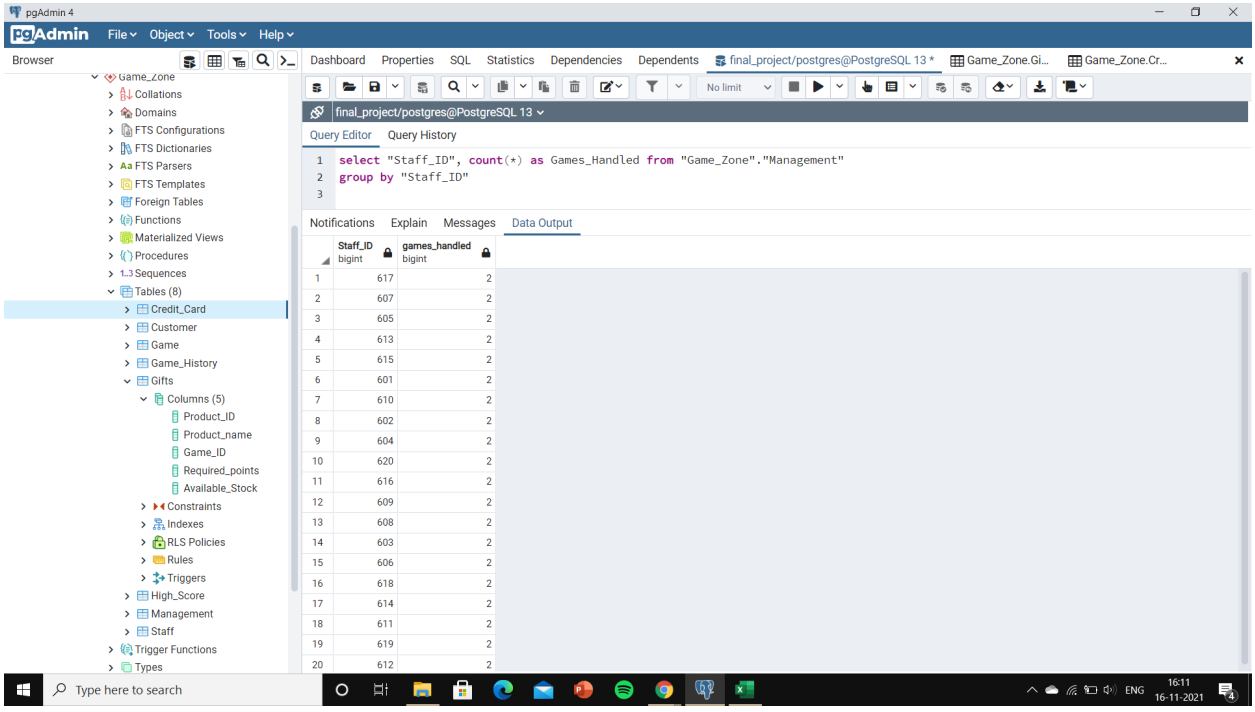
```
select "Game_ID","Required_points"
from "Game_Zone"."Gifts"
where "Gifts"."Game_ID"=405
```



No. of tuples: 1

15. Get the number of games handled by each of the staff member.
SQL query :

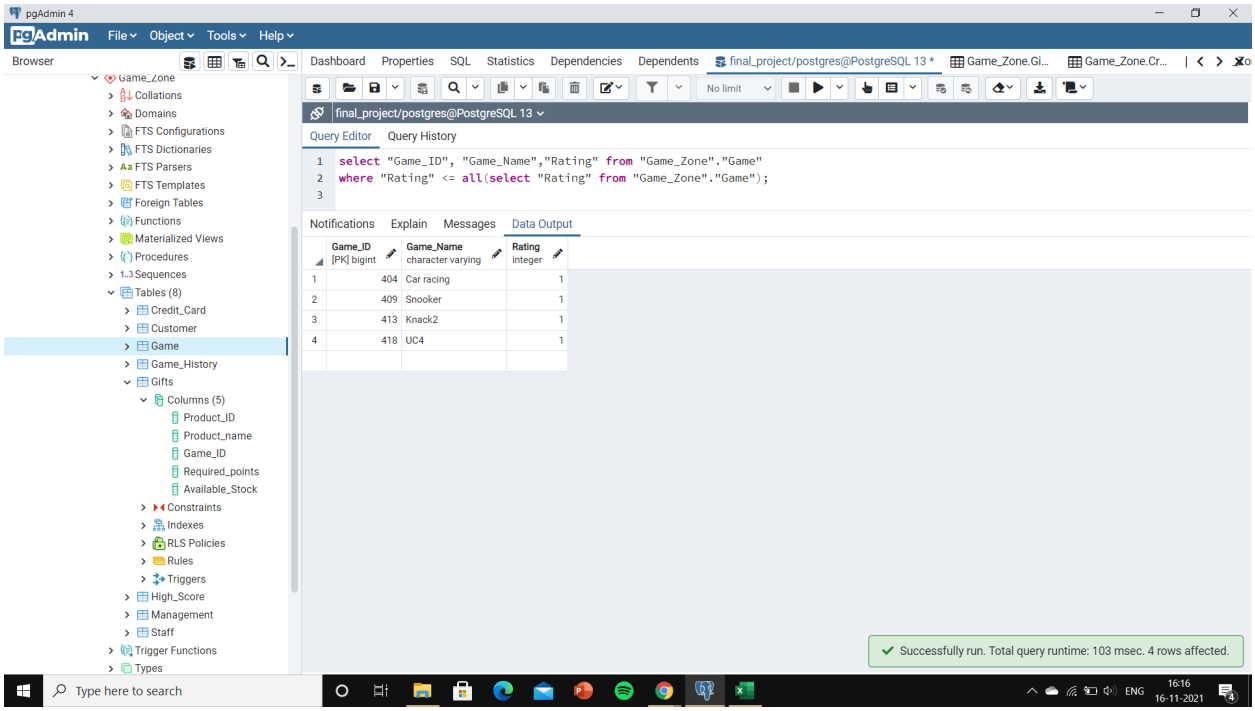
```
select "Staff_ID", count(*) as Games_Handled from
"Game_Zone"."Management"
group by "Staff_ID"
```



No. of Tuples = 20

16. Find the id, name, rating of game which is least rated.
SQL query :

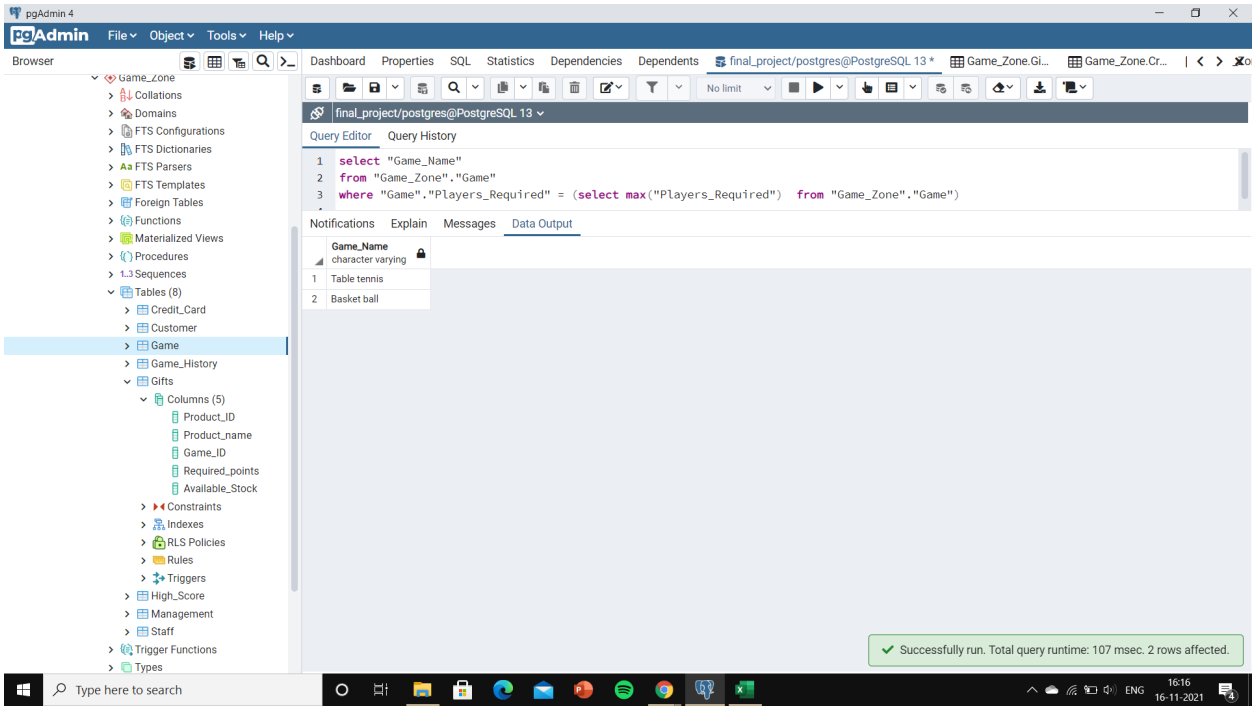
```
select "Game_ID", "Game_Name","Rating" from "Game_Zone"."Game"
where "Rating" <= all(select "Rating" from "Game_Zone"."Game");
```



No. of tuples = 4

17. Find the game which has highest number of players required.
SQL query :

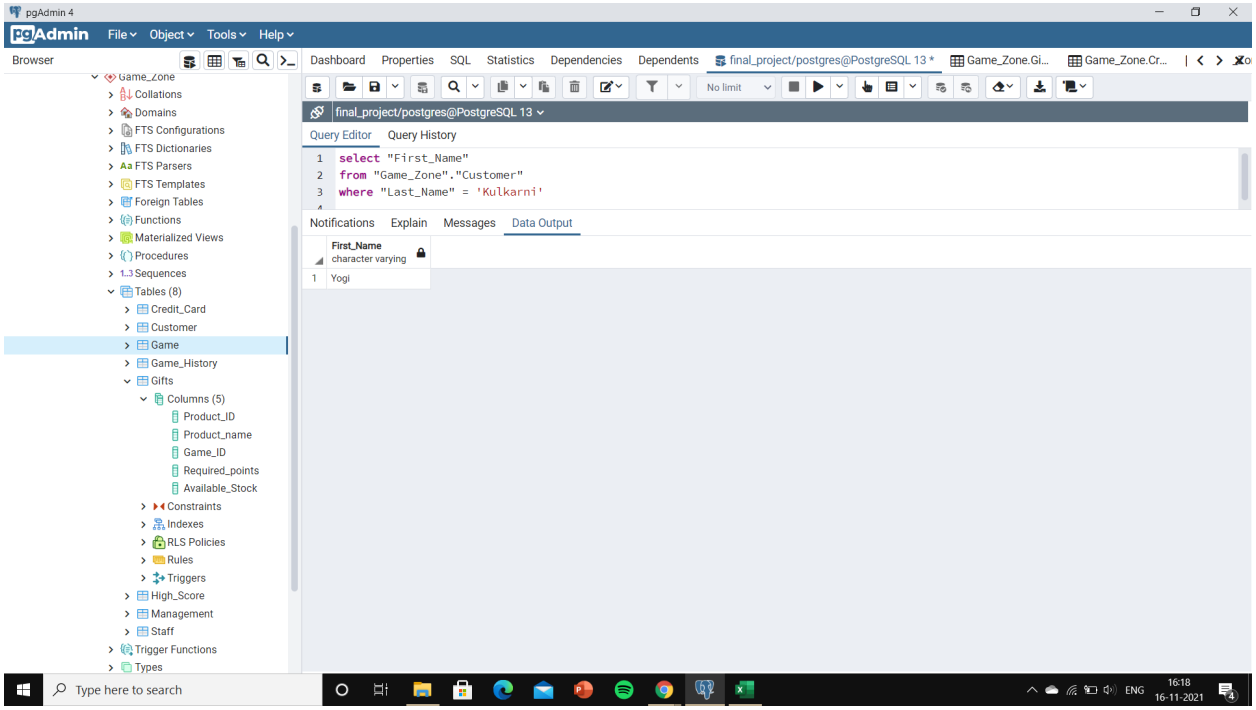
```
select "Game_Name"
from "Game_Zone"."Game"
where "Game"."Players_Required" = (select max("Players_Required")
from "Game_Zone"."Game")
```



No. of Tuples = 2

18. Find the customer's name whose surname is 'Kulkarni.'
SQL query:

```
select "First_Name"
from "Game_Zone"."Customer"
where "Last_Name" = 'Kulkarni'
```

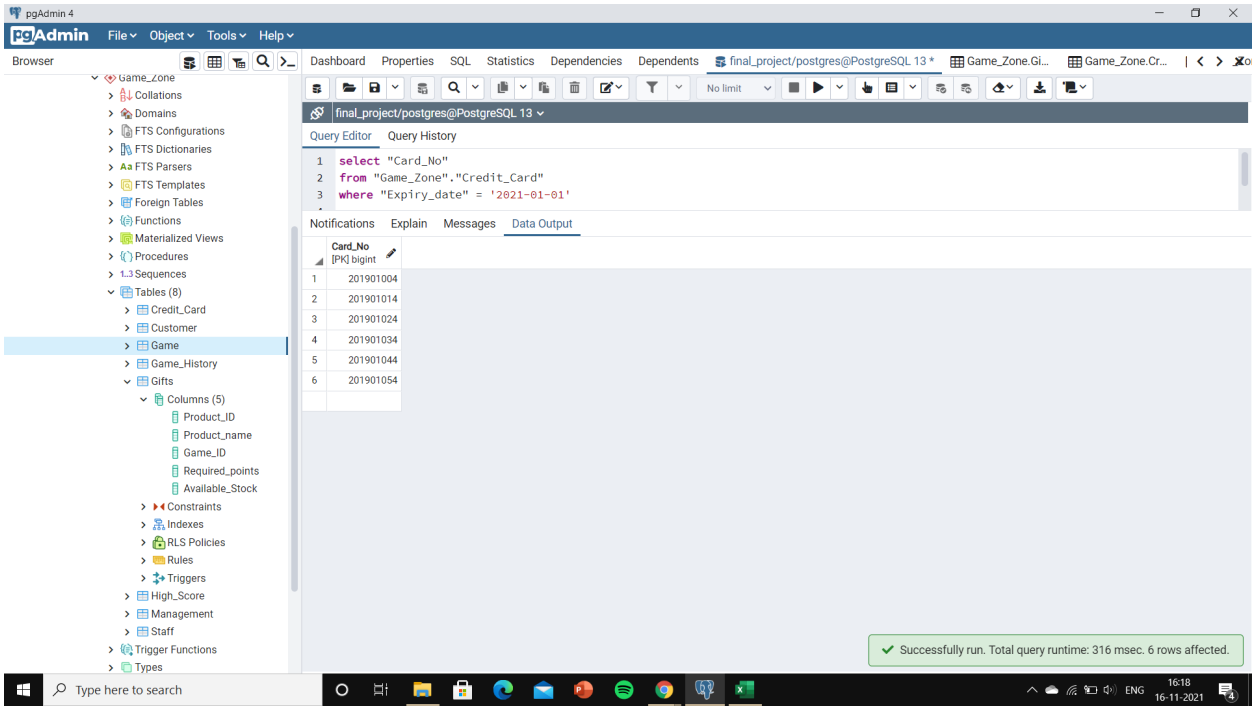


No. of tuples = 1

19. Find the credit card number of customer whose expiry date is 2021-10-01.

SQL query:

```
select "Card_No"
from "Game_Zone"."Credit_Card"
where "Expiry_date" = '2021-01-01'
```

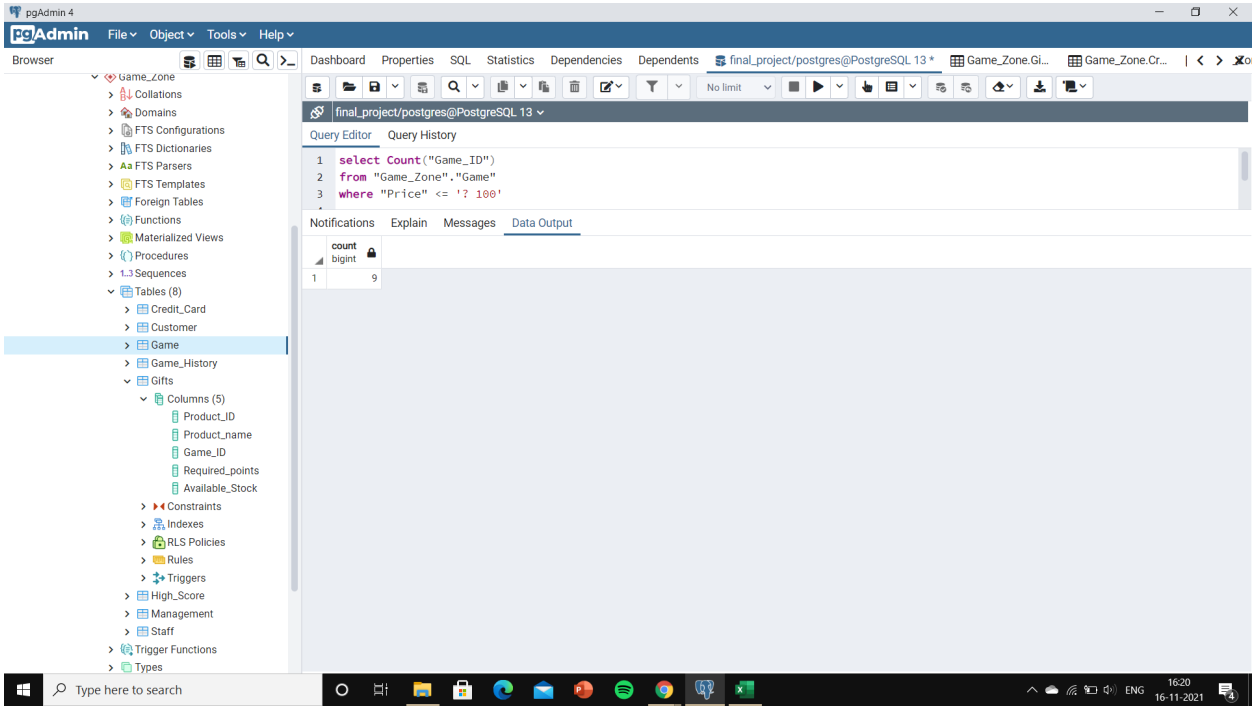


No. of Tuples = 6

20. Count the number of games whose price is less than or equal to 100.

SQL query :

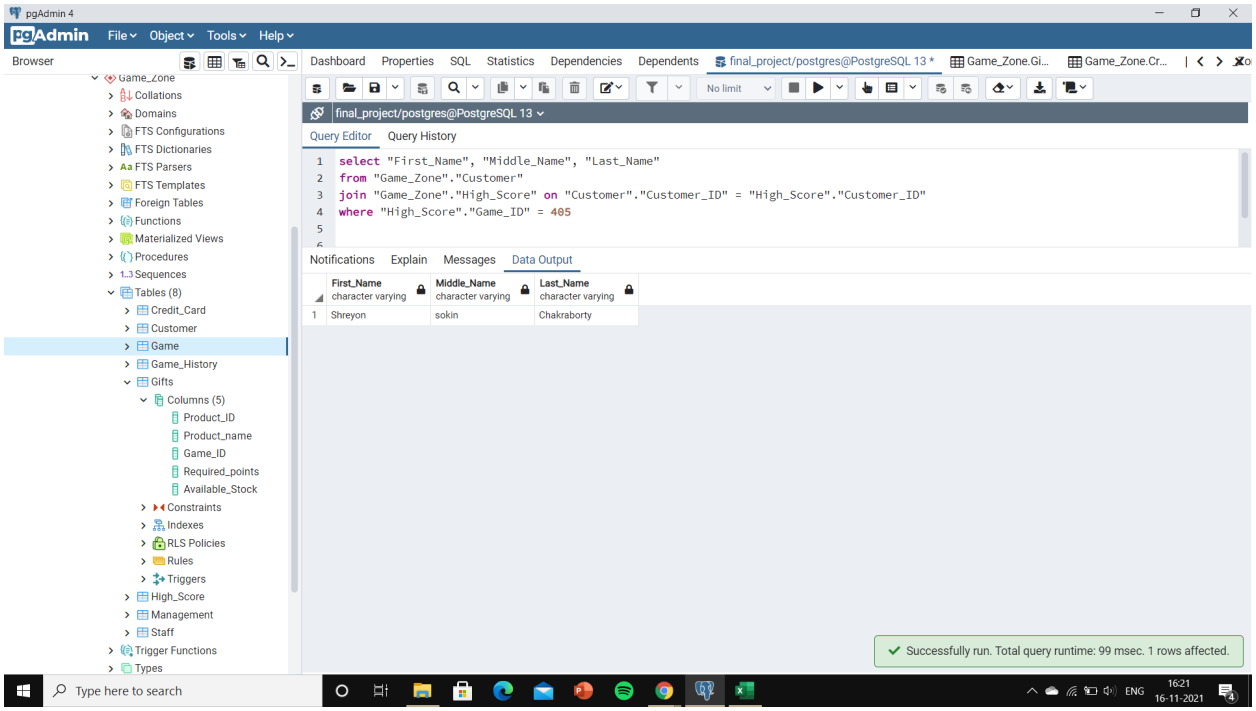
```
select Count("Game_ID")
from "Game_Zone"."Game"
where "Price" <= 100
```



No. of Tuples = 1

21. Find all the names of customers who have achieved the highest score in a game with game_id = 405.
SQL query :

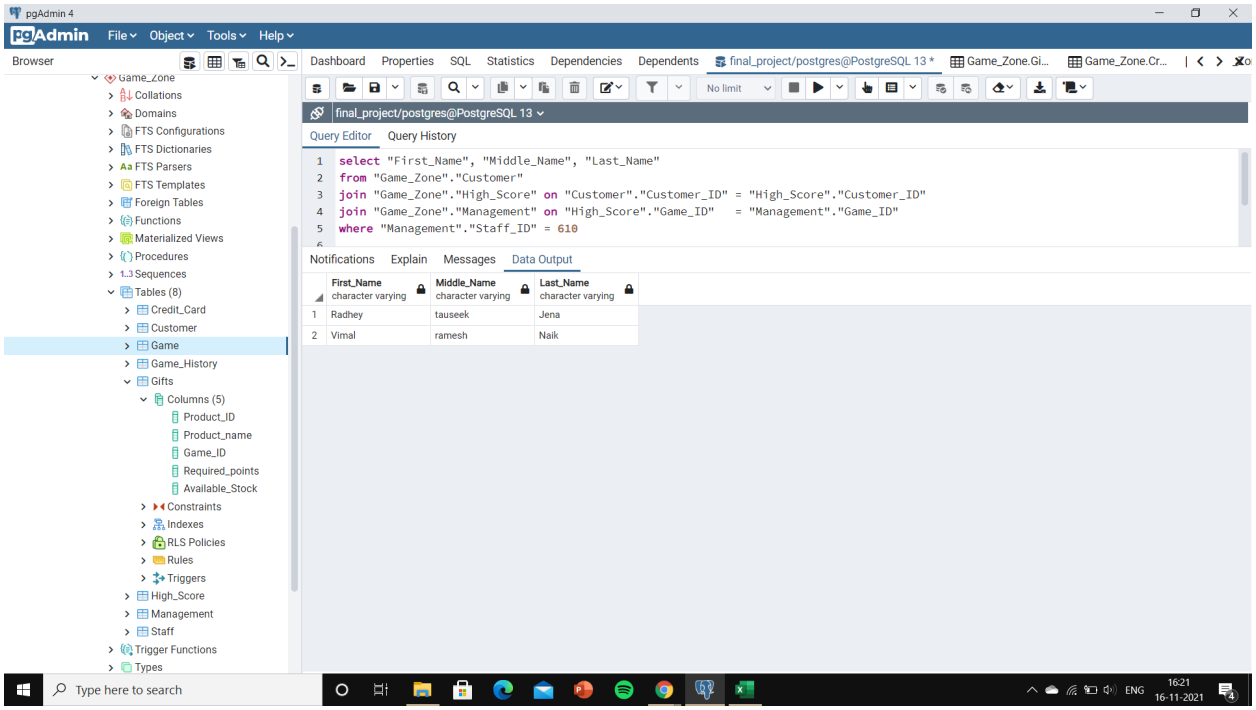
```
select "First_Name", "Middle_Name", "Last_Name"
from "Game_Zone"."Customer"
join "Game_Zone"."High_Score" on "Customer"."Customer_ID" =
"High_Score"."Customer_ID"
where "High_Score"."Game_ID" = 405
```



No. of Tuples = 1

22. Find all the names of customers who have achieved the highest score in a game which is managed by staff_id = 610.
SQL query :

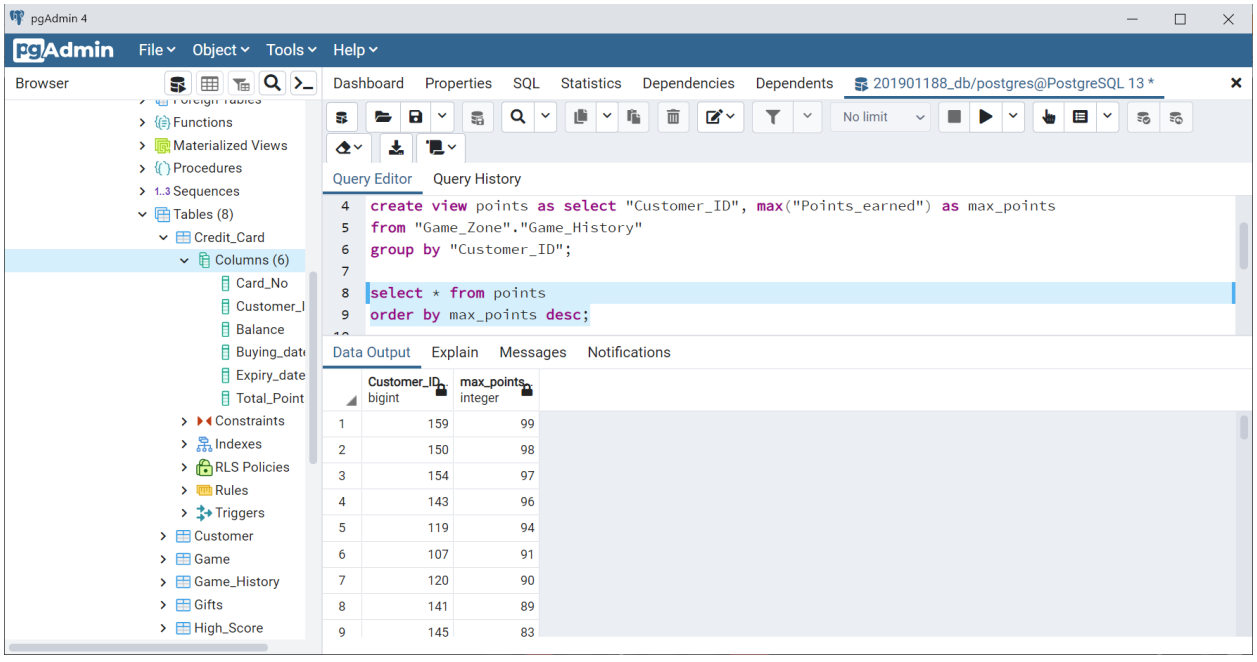
```
select "First_Name", "Middle_Name", "Last_Name"
from "Game_Zone"."Customer"
join "Game_Zone"."High_Score" on "Customer"."Customer_ID" =
"High_Score"."Customer_ID"
join "Game_Zone"."Management" on "High_Score"."Game_ID"    =
"Management"."Game_ID"
where "Management"."Staff_ID" = 610
```



No. of Tuples = 2

23.Get the customer names in the order sorted by the max points scored in any game.
SQL query :

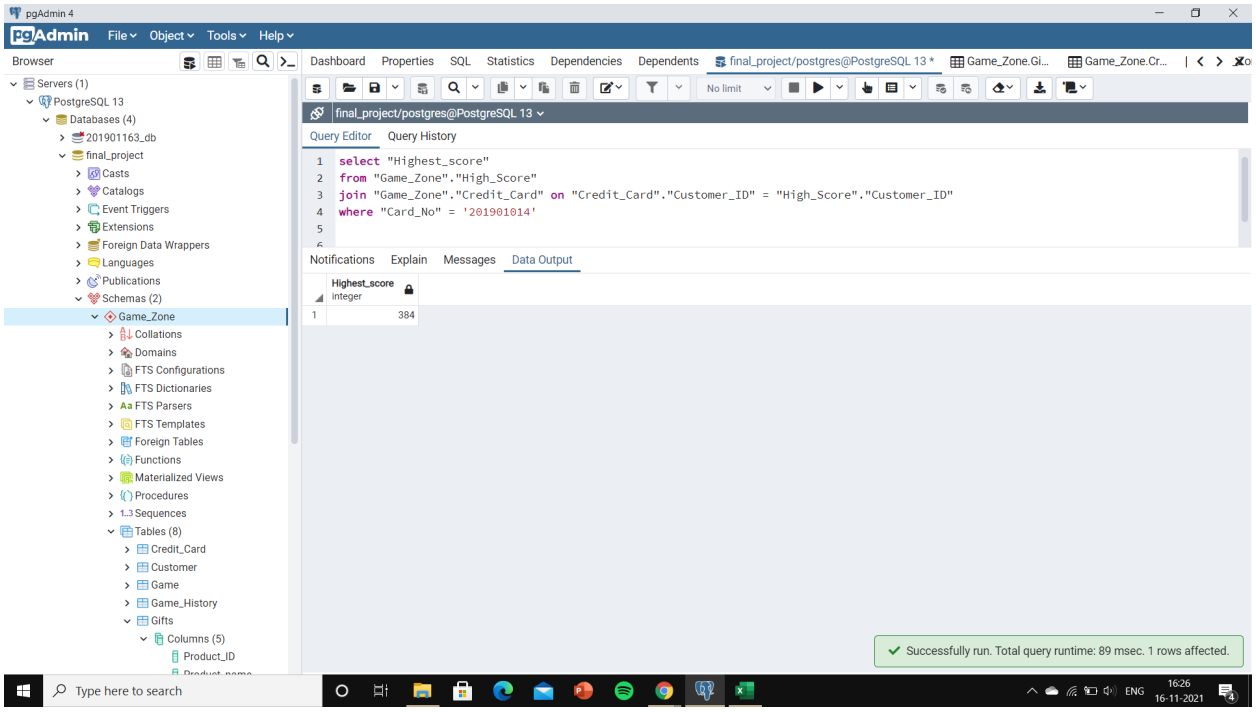
```
create view points as select "Customer_ID", max("Points_earned") as max_points
from "Game_Zone"."Game_History"
group by "Customer_ID";
select * from points
order by max_points desc;
```



No of Tuples = 60

24. Find the highest score of the customer whose credit card number is 201901014.
SQL query :

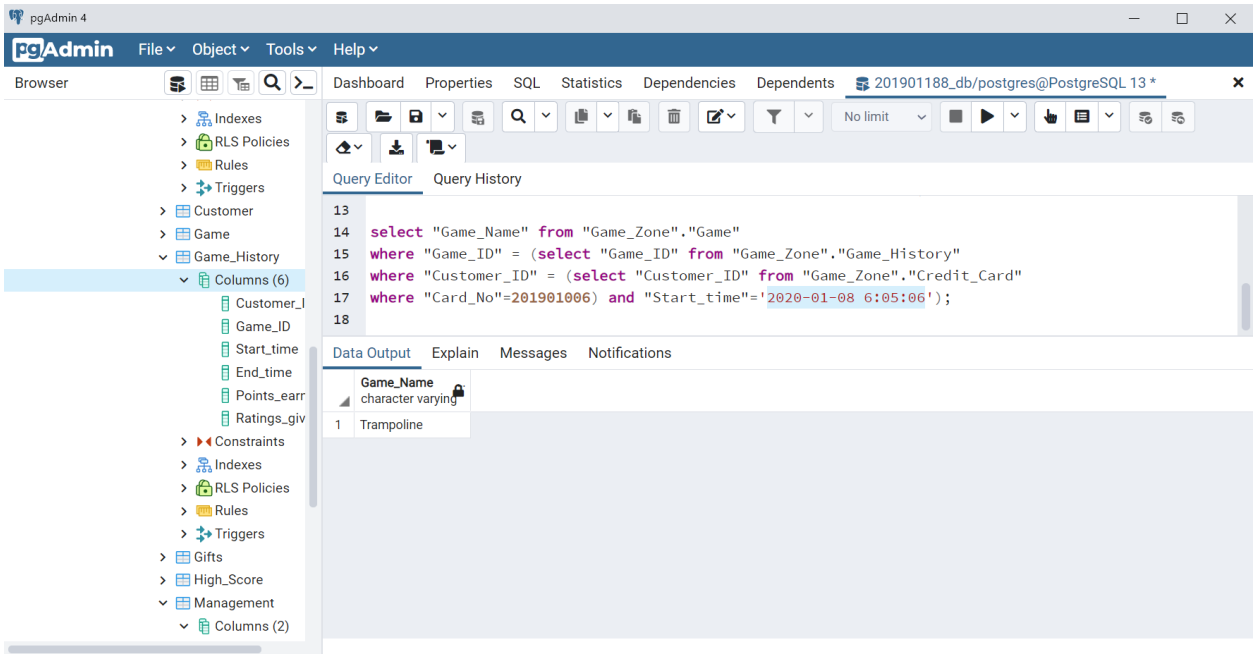
```
select "Highest_score"
from "Game_Zone"."High_Score"
join "Game_Zone"."Credit_Card" on "Credit_Card"."Customer_ID" =
"High_Score"."Customer_ID"
where "Card_No" = '201901014'
```



No. of tuples = 1

25. Find the name of the game played by the customer whose card_no is 201901006 at time 2020-01-08 6:05:06.
SQL query :

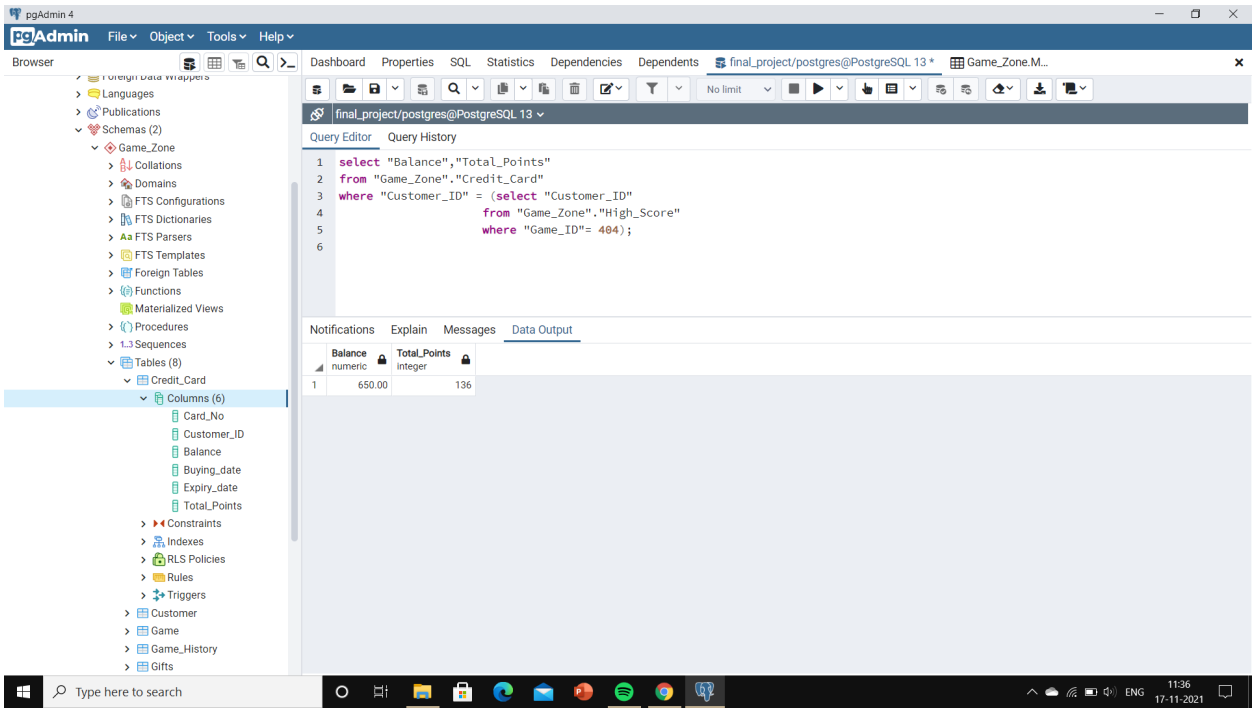
```
select "Game_Name" from "Game_Zone"."Game"
where "Game_ID" = (select "Game_ID" from "Game_Zone"."Game_History"
where "Customer_ID" = (select "Customer_ID" from "Game_Zone"."Credit_Card"
where "Card_No"=201901006) and "Start_time"='2020-01-08 6:05:06');
```



No. of Tuples = 1

26. Find the customer balance and earned point who score highest in game with game_id 404.
SQL query :

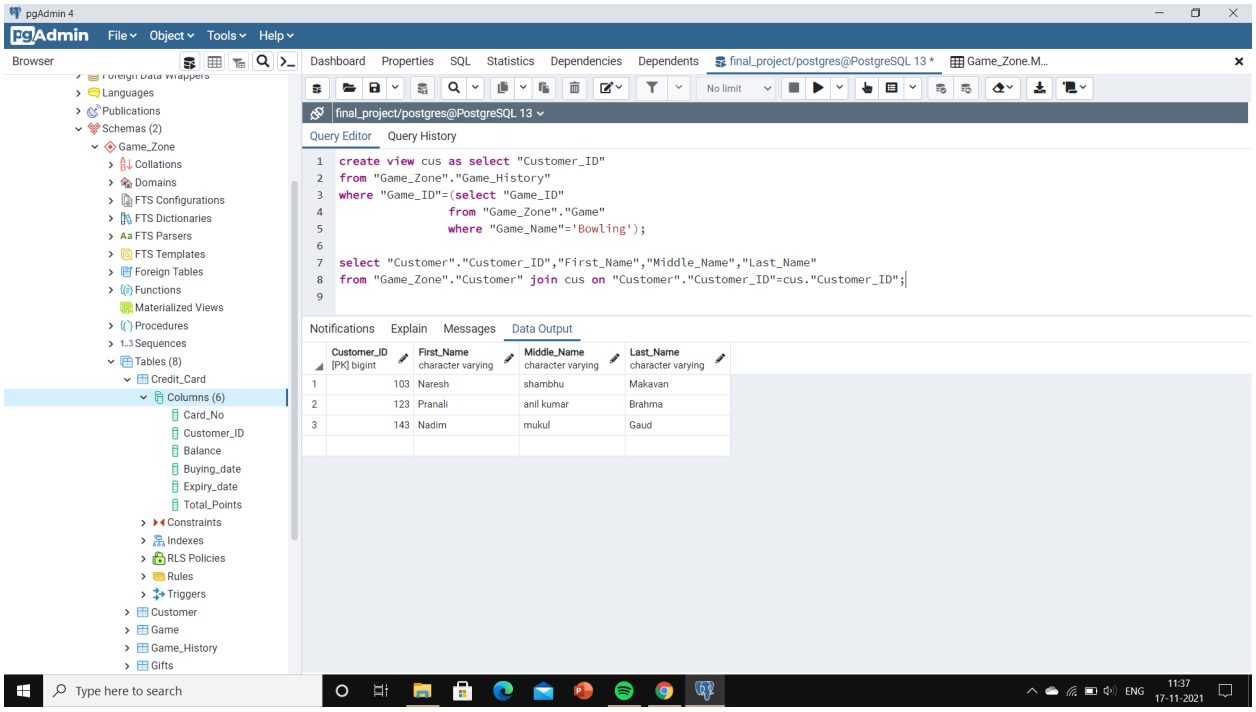
```
select "Balance","Total_Points"
from "Game_Zone"."Credit_Card"
where "Customer_ID" = (select "Customer_ID"
                       from "Game_Zone"."High_Score"
                       where "Game_ID"= 404);
```



No. of Tuples = 1

27.Find customers' names and id who played the game Bowling .
SQL query :

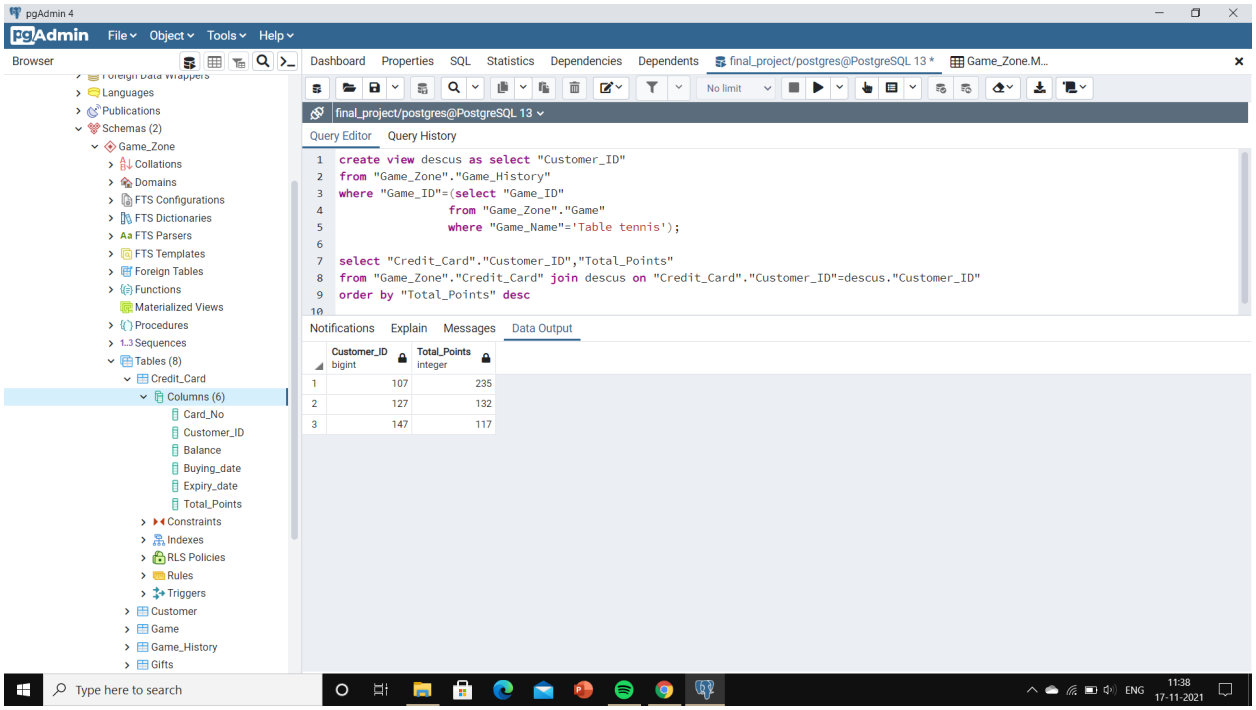
```
create view cus as select "Customer_ID"
from "Game_Zone"."Game_History"
where "Game_ID"=(select "Game_ID"
                        from "Game_Zone"."Game"
                        where "Game_Name"='Bowling');

select "Customer"."Customer_ID","First_Name","Middle_Name","Last_Name"
from "Game_Zone"."Customer" join cus on
"Customer"."Customer_ID"=cus."Customer_ID";
```



No. of Tuples = 3

28. Find the customers' names and total points they earned in descending order who played the game Table tennis.
SQL query :

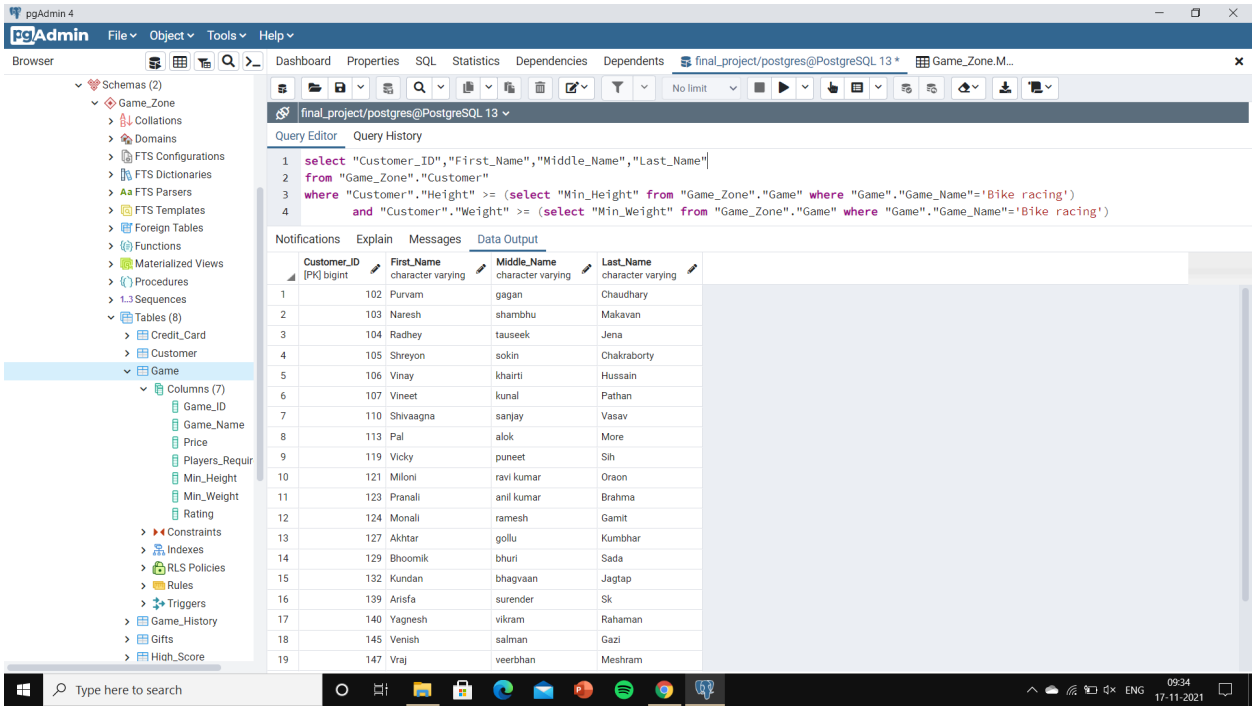
```
create view descus as select "Customer_ID"
from "Game_Zone"."Game_History"
where "Game_ID"=(select "Game_ID"
                        from "Game_Zone"."Game"
                        where "Game_Name"='Table tennis');

select "Credit_Card"."Customer_ID","Total_Points"
from "Game_Zone"."Credit_Card" join descus on
"Credit_Card"."Customer_ID"=descus."Customer_ID"
order by "Total_Points" desc
```



No. of Tuples = 3

29. Find the customer id and their names who are eligible for bike racing.
SQL query:

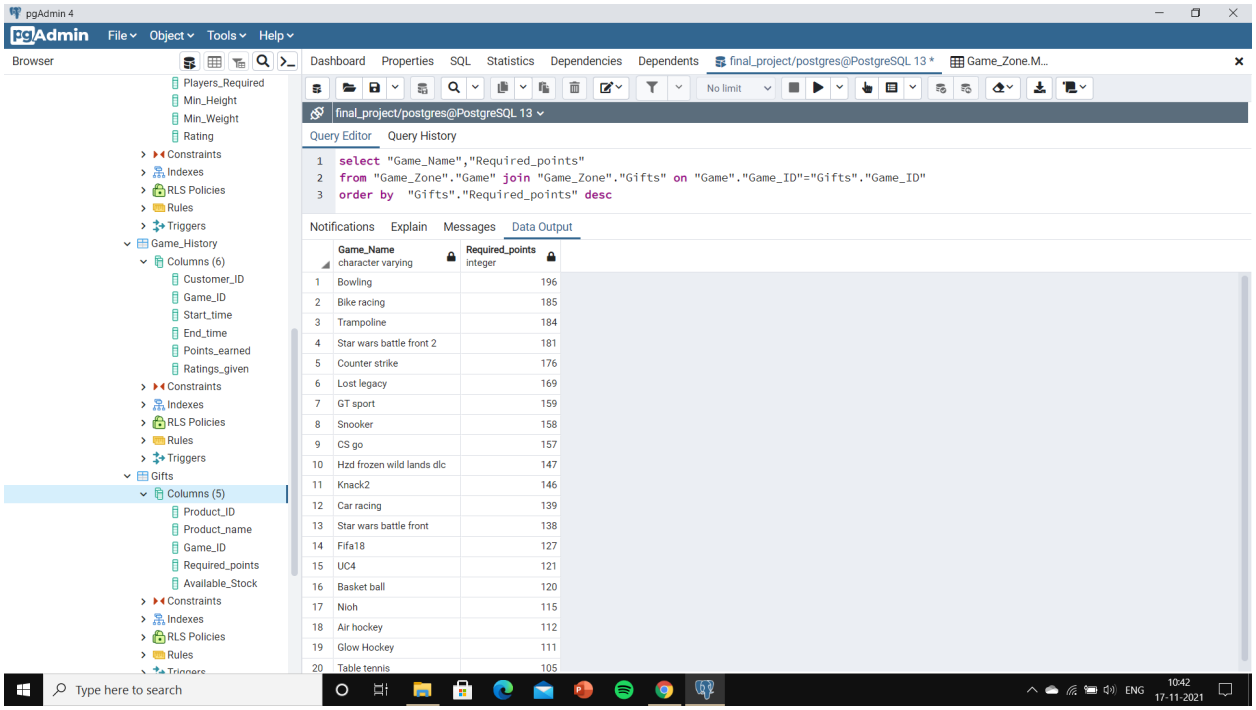
```
select "Customer_ID","First_Name","Middle_Name","Last_Name"
from "Game_Zone"."Customer"
where "Customer"."Height" >= (select "Min_Height" from "Game_Zone"."Game" where
"Game"."Game_Name"='Bike racing')
         and "Customer"."Weight" >= (select "Min_Weight" from
"Game_Zone"."Game" where "Game"."Game_Name"='Bike racing')
```



No. of Tuples = 22

30. List the game's name in descending order of required points to win a gift for a particular game.
SQL query:

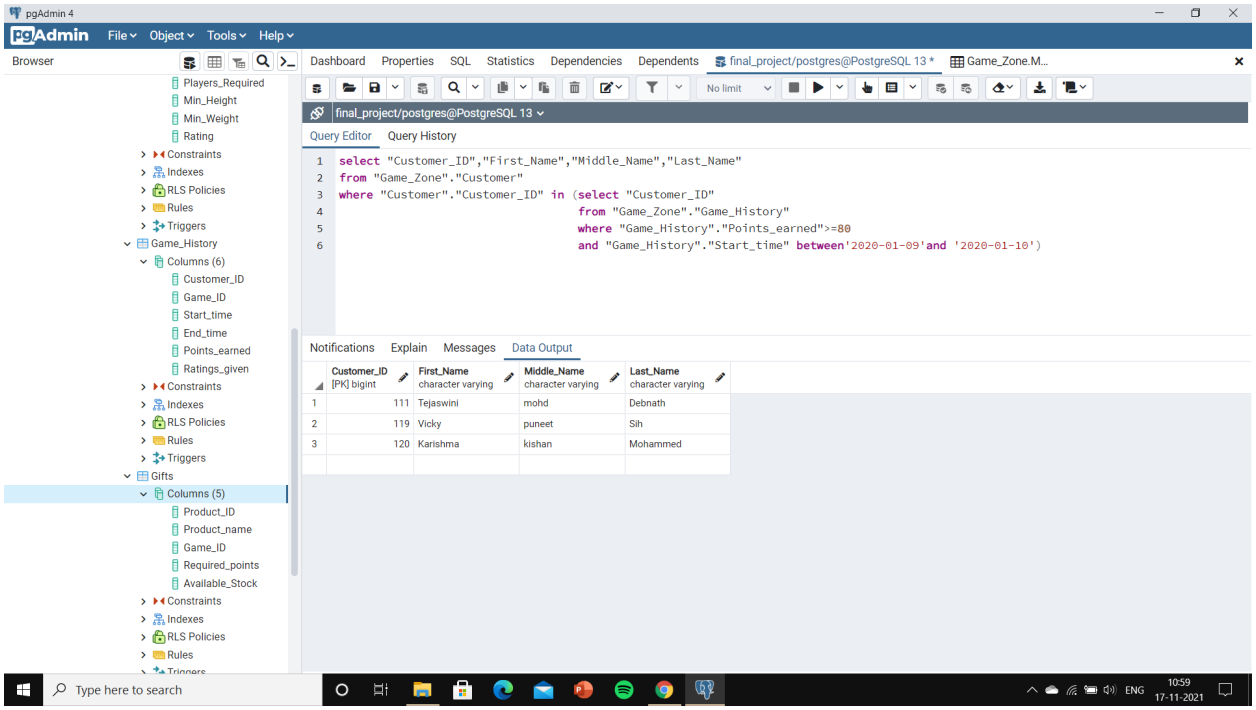
```
select "Game_Name","Required_points"
from "Game_Zone"."Game" join "Game_Zone"."Gifts" on
"Game"."Game_ID"="Gifts"."Game_ID"
order by  "Gifts"."Required_points" desc
```



No. of Tuples = 20

31. Find the customer's id and name who has earned greater than 80 points on date 2020-01-09.
SQL query:

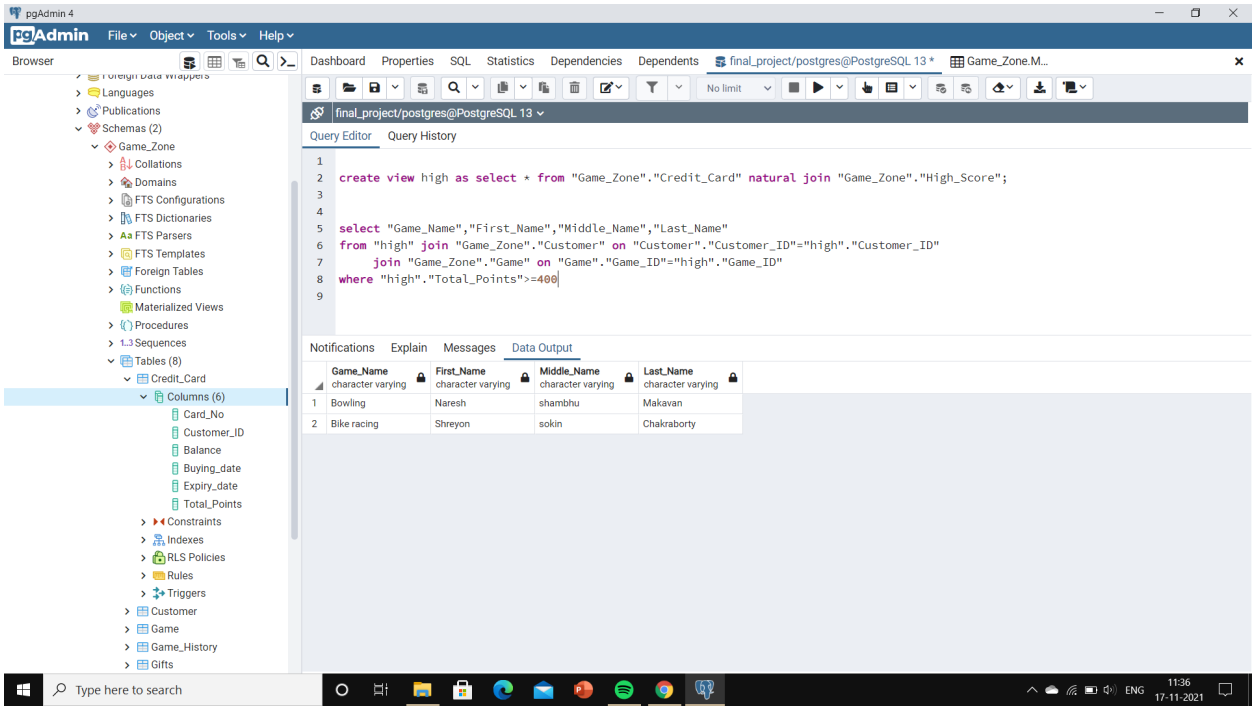
```
Select "Customer_ID","First_Name","Middle_Name","Last_Name"
from "Game_Zone"."Customer"
where "Customer"."Customer_ID"
                        in  (select "Customer_ID"
                               from "Game_Zone"."Game_History"
                               where    "Game_History"."Points_earned">=80
                              and "Game_History"."Start_time"
                              between'2020-01-09'and '2020-01-10')
```



No. of Tuples = 3

32. Find the game name and customer's name who has at least 400 total points and has a high score in any game.
SQL query:

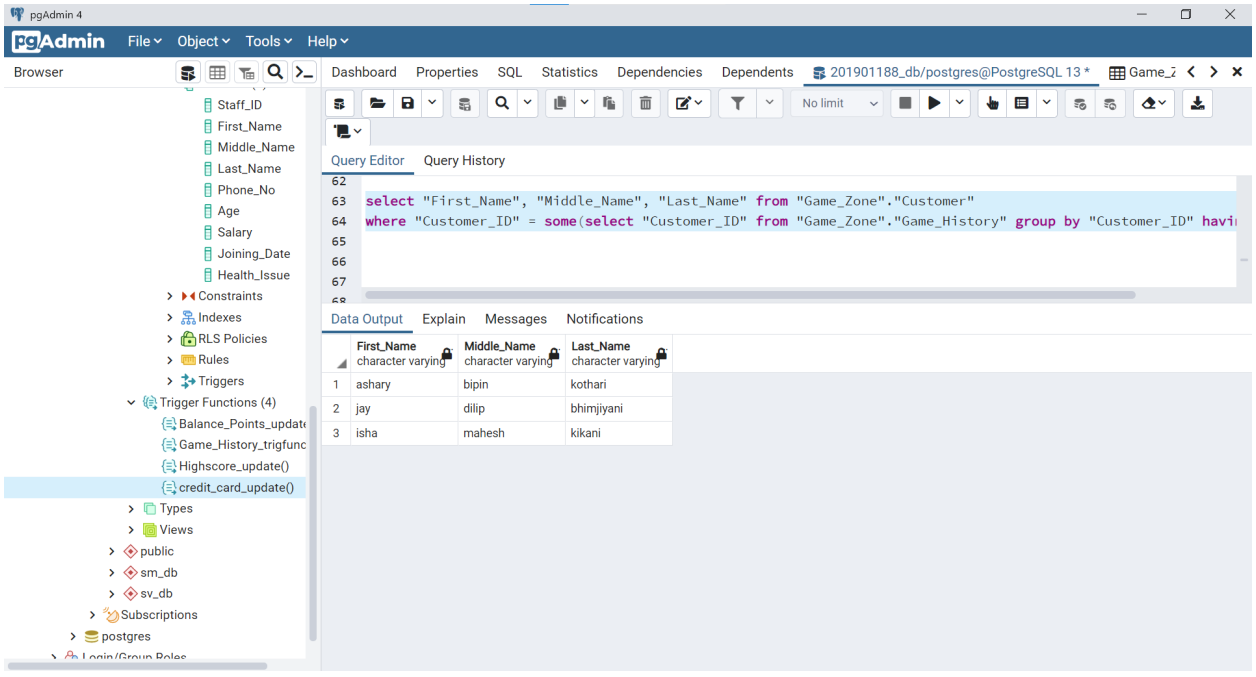
```
create view high as select * from "Game_Zone"."Credit_Card" natural join
"Game_Zone"."High_Score";

select "Game_Name","First_Name","Middle_Name","Last_Name"
from "high" join "Game_Zone"."Customer" on
"Customer"."Customer_ID"="high"."Customer_ID"
       join "Game_Zone"."Game" on "Game"."Game_ID"="high"."Game_ID"
where "high"."Total_Points">=400
```



No. of Tuples = 20

33. Get the names of the customers who have played games more than once.
SQL query :

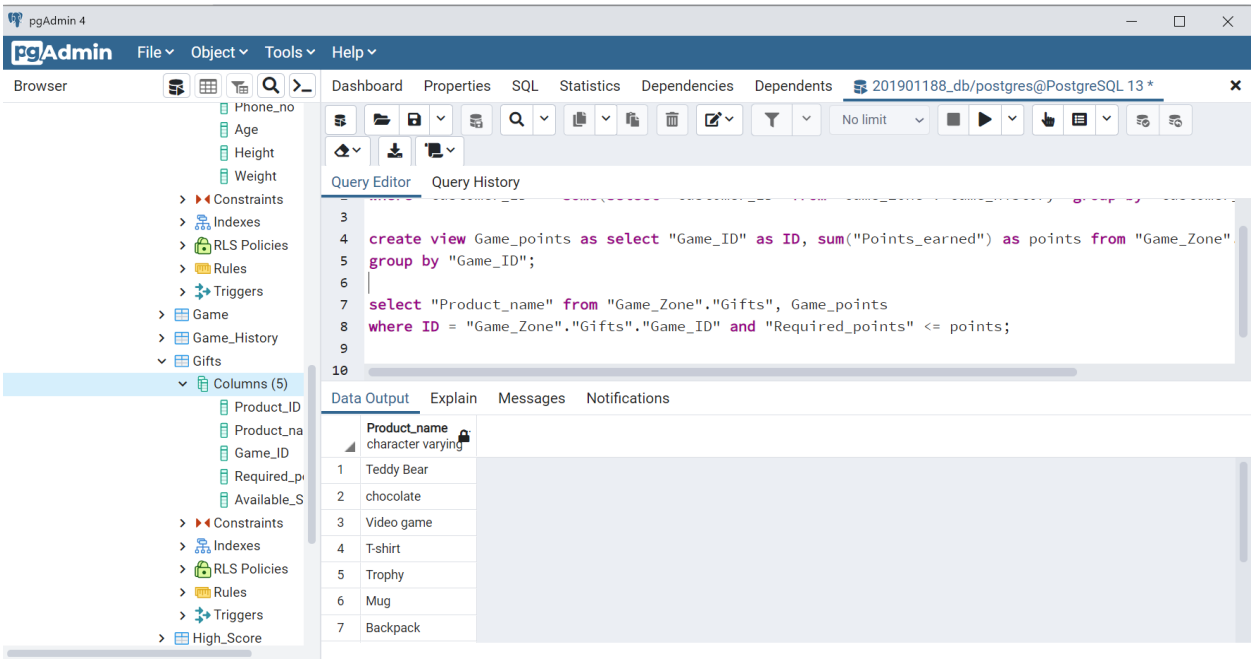
```
select "First_Name", "Middle_Name", "Last_Name" from "Game_Zone"."Customer"
where "Customer_ID" = some(select "Customer_ID" from "Game_Zone"."Game_History"
group by "Customer_ID" having count(*)>1);
```



No of Tuples = 3

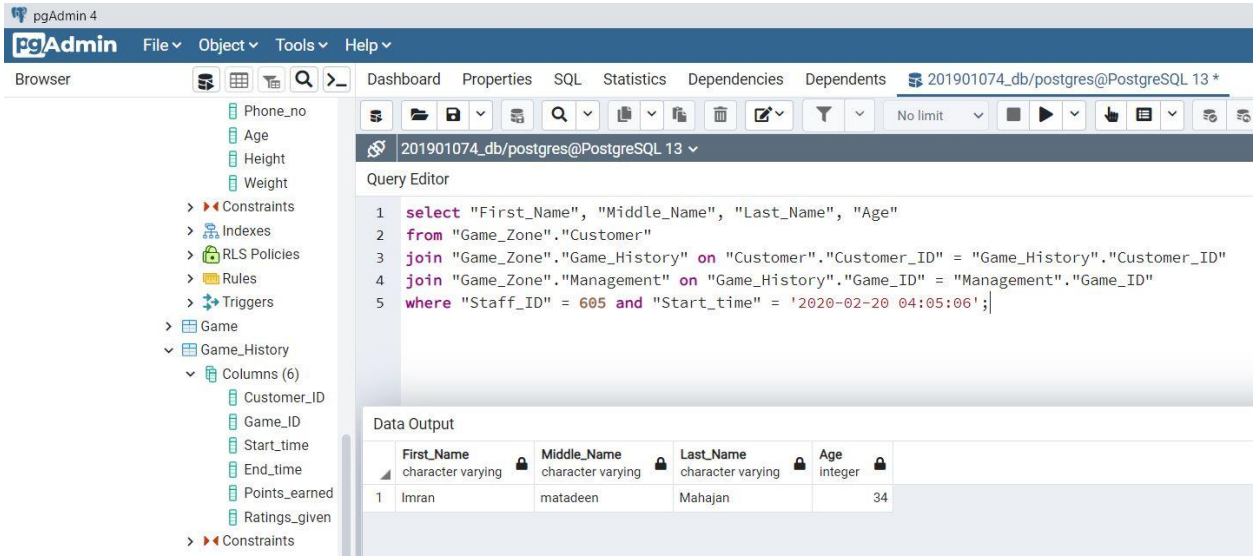
34. List the names of all the gifts which can be bought by at least one of the customers.
SQL query :

```
create view Game_points as select "Game_ID" as ID, sum("Points_earned") as
points from "Game_Zone"."Game_History"
group by "Game_ID";
select "Product_name" from "Game_Zone"."Gifts", Game_points
where ID = "Game_Zone"."Gifts"."Game_ID" and "Required_points" <= points;
```



35. What is the name and age of the customer who was playing a game which is managed by staff_id = 605 at time '2020-02-20 04:05:06'?
SQL query :

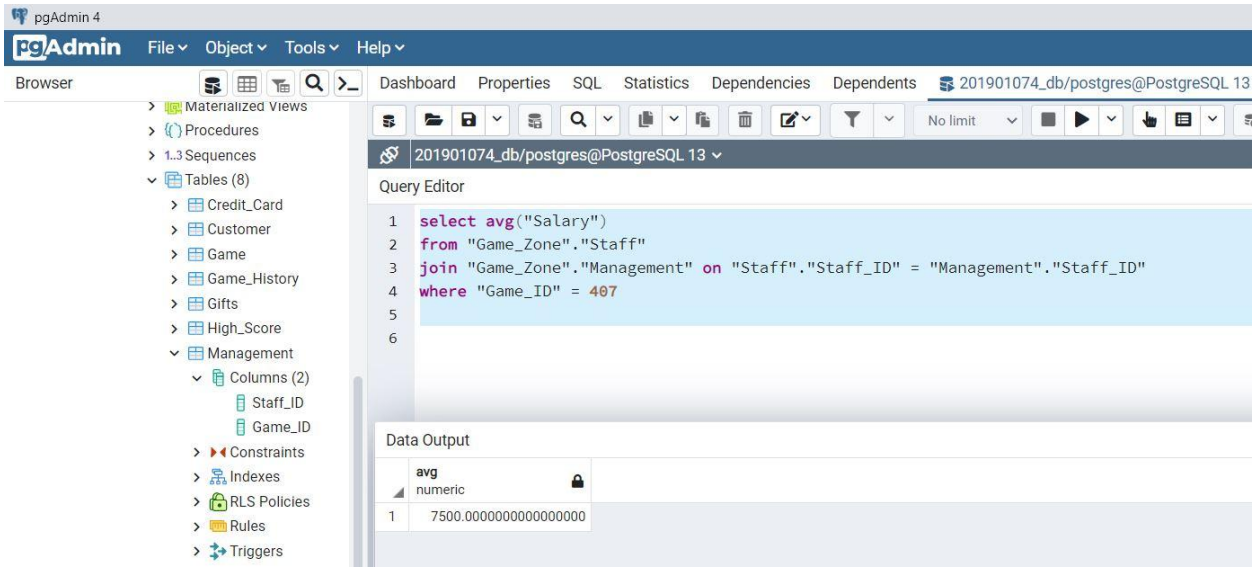
```
select "First_Name", "Middle_Name", "Last_Name", "Age"
from "Game_Zone"."Customer"
join "Game_Zone"."Game_History" on "Customer"."Customer_ID" =
"Game_History"."Customer_ID"
join "Game_Zone"."Management" on "Game_History"."Game_ID" =
"Management"."Game_ID"
where "Staff_ID" = 605 and "Start_time" = '2020-02-20 04:05:06';
```



No of Tuples =1

36. Find out the average salary of the staff members who manage the game with game_id = 407
SQL query :

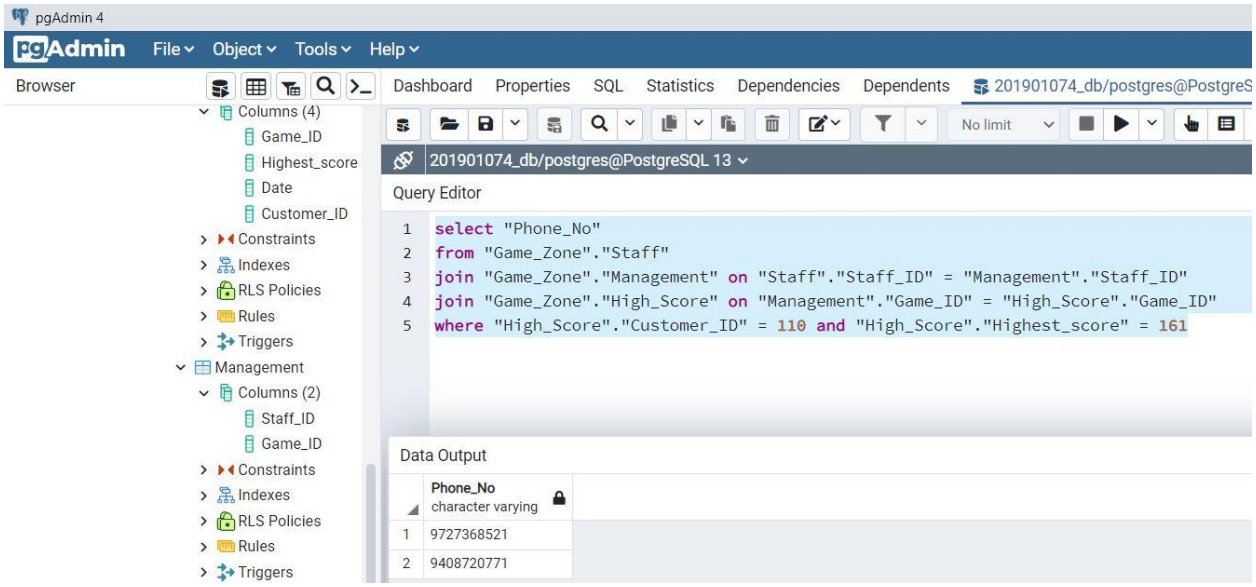
```
select avg("Salary")
from "Game_Zone"."Staff"
join "Game_Zone"."Management" on "Staff"."Staff_ID" = "Management"."Staff_ID"
where "Game_ID" = 407
```



No of Tuples =1

37. Find the phone numbers of all the staff members who manage the game in which the customer with customer_id = 110 has the highest score.
SQL query :

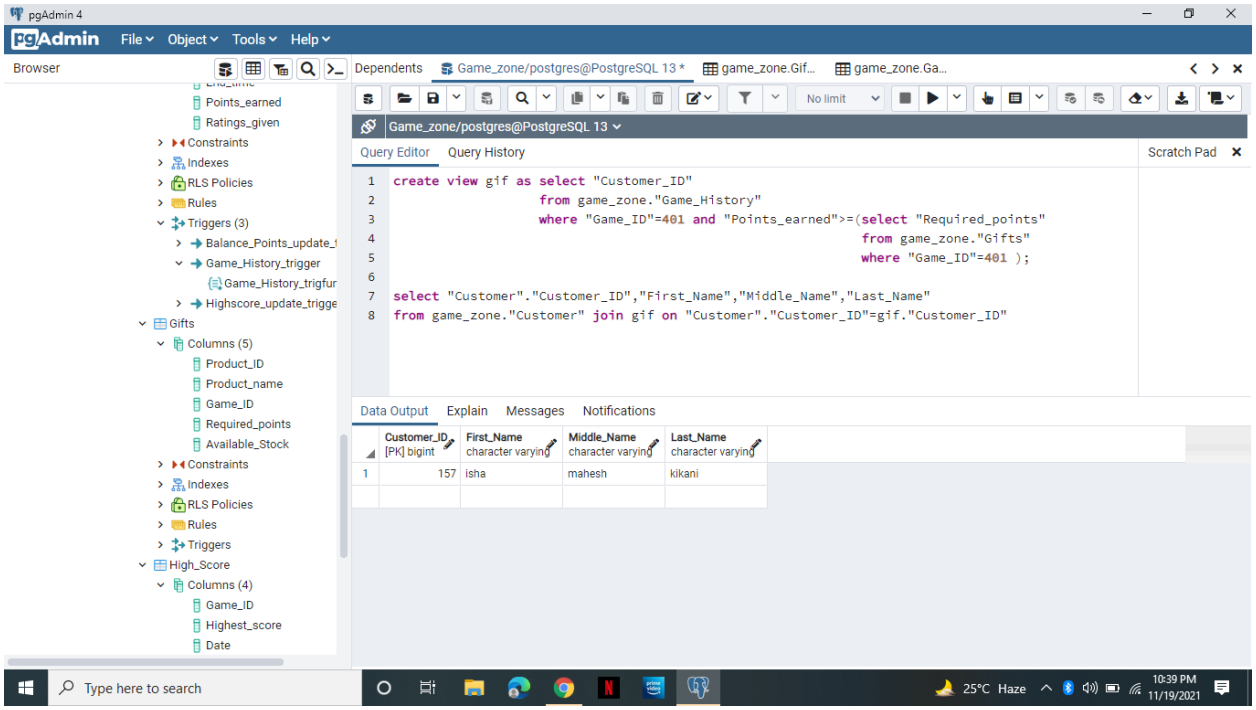
```
select "Phone_No"
from "Game_Zone"."Staff"
join "Game_Zone"."Management" on "Staff"."Staff_ID" = "Management"."Staff_ID"
join "Game_Zone"."High_Score" on "Management"."Game_ID" =
"High_Score"."Game_ID"
where "High_Score"."Customer_ID" = 110 and "High_Score"."Highest_score" = 161
```



No of Tuples = 2

38. Find Customers name and id who receive gift in game which has game id 401.
SQL query :

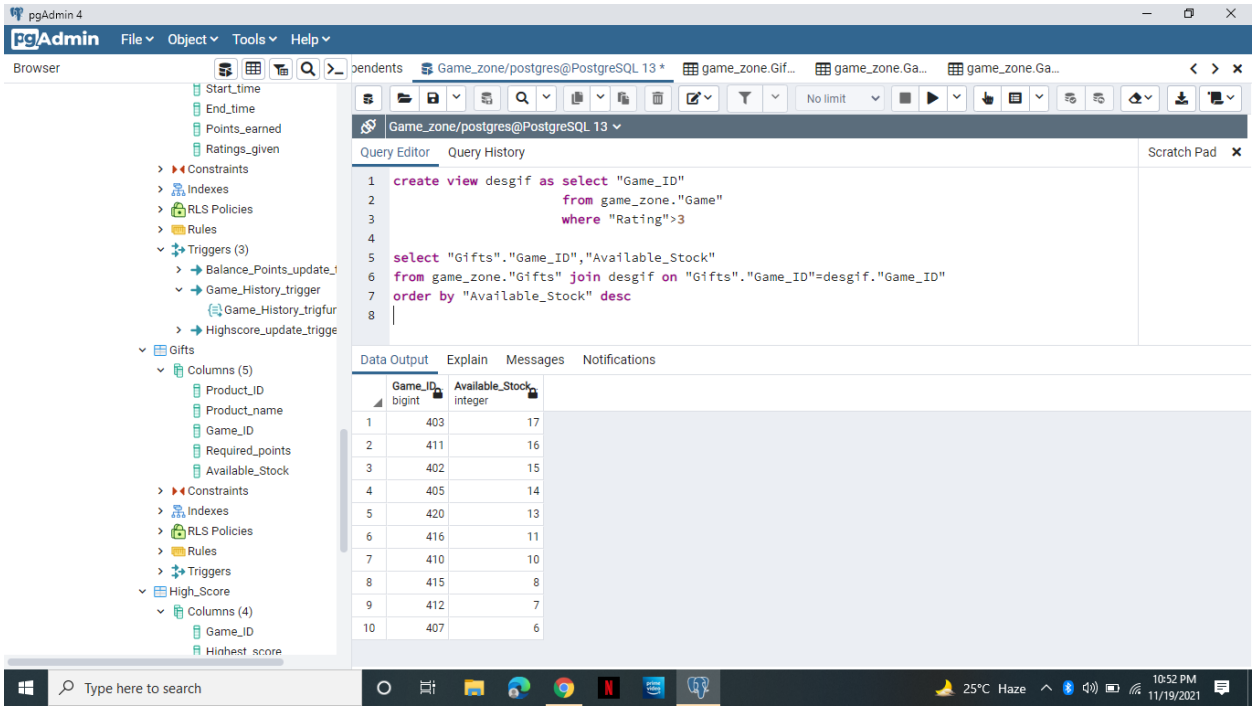
```
create view gif as select "Customer_ID"
                  from game_zone."Game_History"
                          where "Game_ID"=401 and "Points_earned">=(select
"Required_points"
                        from game_zone."Gifts"
                         where "Game_ID"=401 );

select
"Customer"."Customer_ID","First_Name","Middle_Name","Last_Name"
from game_zone."Customer" join gif on
"Customer"."Customer_ID"=gif."Customer_ID"
```



No of Tuples =1

39. List the games in decreasing order of their available stock which has rating greater than 3 .
SQL query :

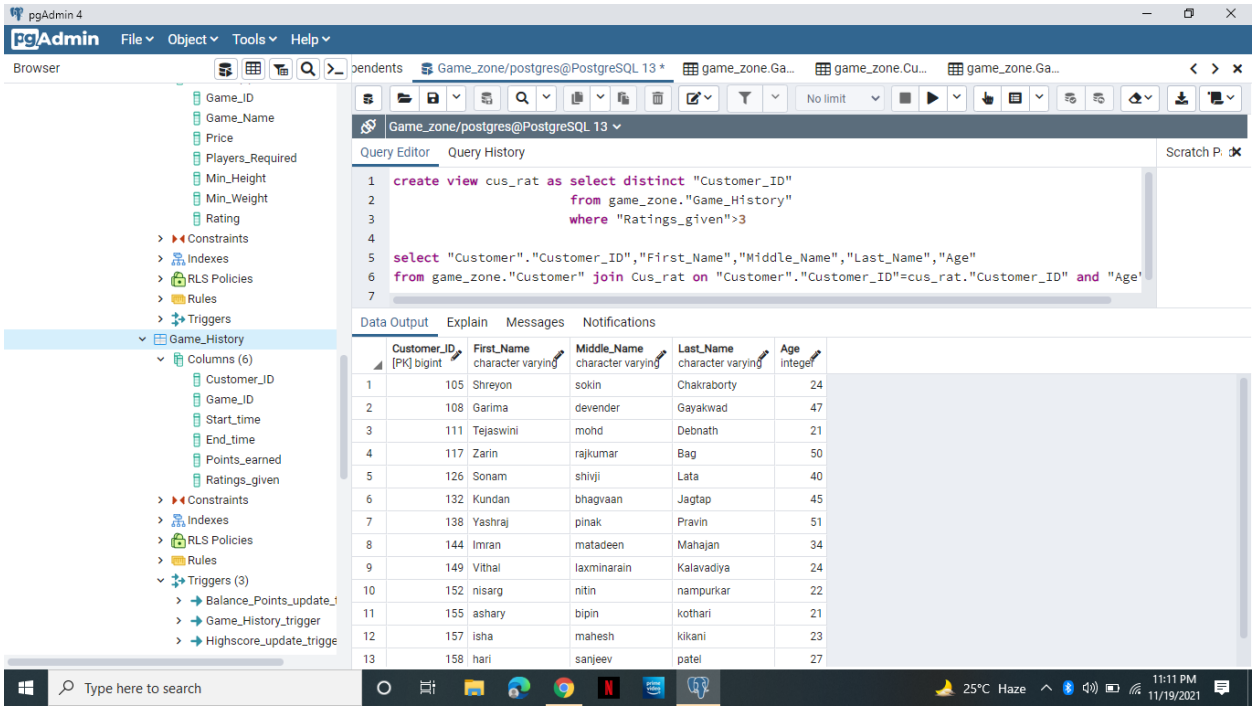
```
create view desgif as select "Game_ID"
                                from game_zone."Game"
                                where "Rating">3

select "Gifts"."Game_ID","Available_Stock"
from game_zone."Gifts" join desgif on "Gifts"."Game_ID"=desgif."Game_ID"
order by "Available_Stock" desc
```



No of Tuples = 10

40. List the customer who is given rating greater than 3 and their age is greater than 20.
SQL query :

```
create view cus_rat as select distinct "Customer_ID"
                                from game_zone."Game_History"
                                where "Ratings_given">3

select "Customer"."Customer_ID","First_Name","Middle_Name","Last_Name","Age"
from game_zone."Customer" join Cus_rat on
"Customer"."Customer_ID"=cus_rat."Customer_ID" and "Age">20
```



No of Tuples = 13

Trigger Function:

1.Trigger for updating rating of games

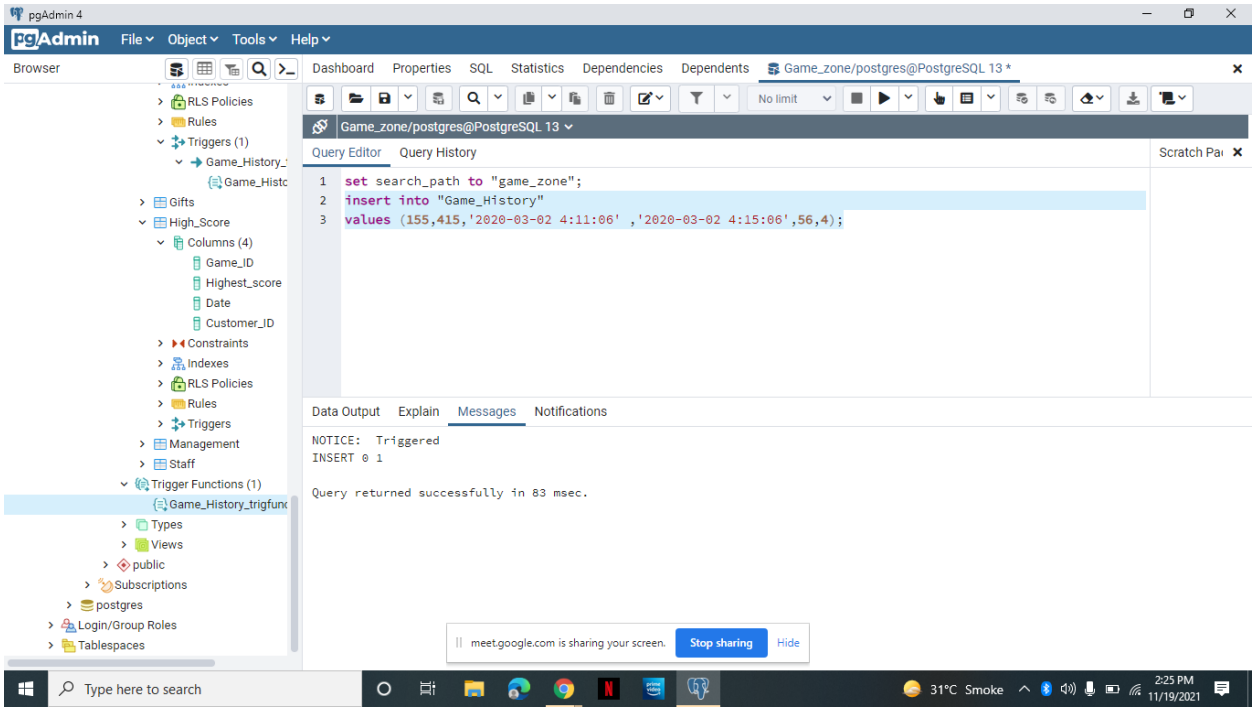
Trigger function SQL code:

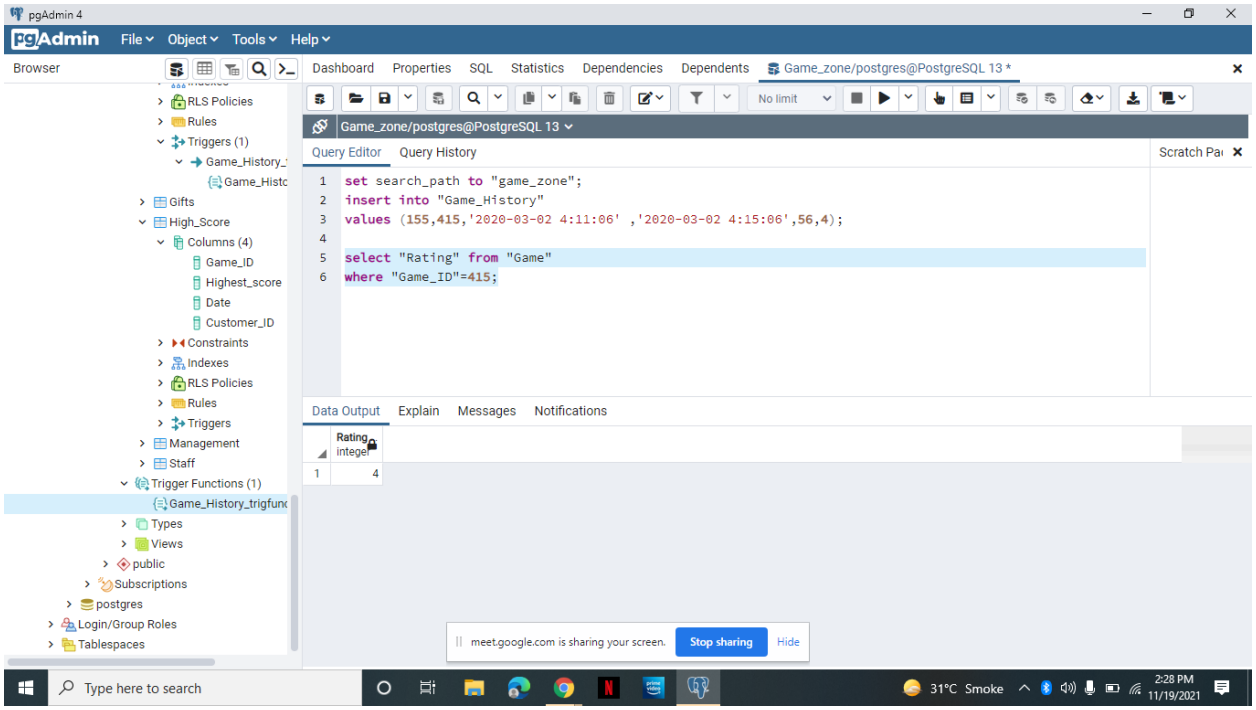
```
CREATE FUNCTION game_zone."Game_History_trigfunc"()
    RETURNS trigger
    LANGUAGE 'plpgsql'
     NOT LEAKPROOF
AS $BODY$
declare
tmp bigint;

begin
tmp=new."Game_ID";
update "Game_Zone"."Game"
set "Rating"=(select avg("Ratings_given") as avg_ratings
                    from "Game_Zone"."Game_History"
                    where "Game_ID"=tmp)
where "Game"."Game_ID"=tmp;
raise notice 'Triggered';
```

```
return NULL;
end
$BODY$;

ALTER FUNCTION game_zone."Game_History_trigfunc"()
    OWNER TO postgres;
```

Creating trigger in table:

```
CREATE TRIGGER "Game_History_trigger"
    AFTER INSERT
    ON game_zone."Game_History"
    FOR EACH ROW
    EXECUTE FUNCTION game_zone."Game_History_trigfunc"();
```





Before inserting in game_history, the rating of the game with id '415' was 1. After inserting, the rating of that game changed to 4, which is displayed in the above screen shot.

2. Trigger for updating balance and total points in credit card

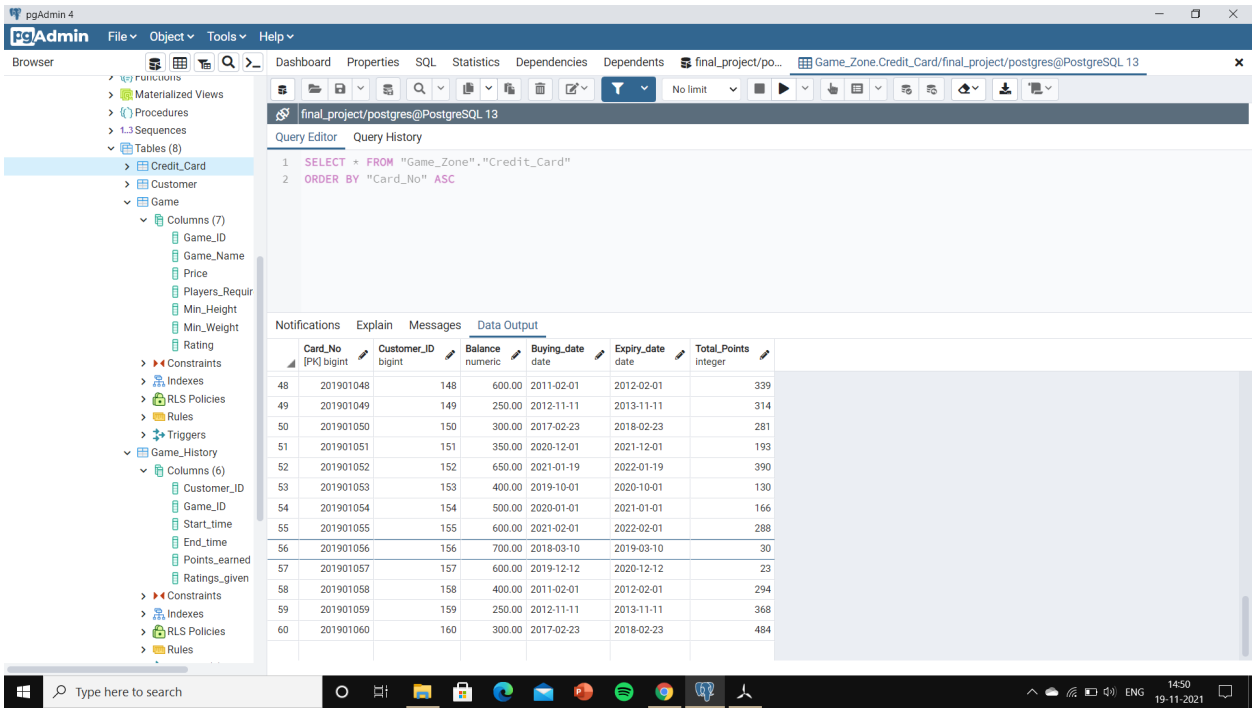
Trigger function SQL code:

```
CREATE OR REPLACE FUNCTION "Game_Zone"."Balance_Points_update"()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    VOLATILE
    COST 100
AS $BODY$
declare
tmp bigint;
x money;
begin
tmp=new."Game_ID";

select "Price" into x from "Game_Zone"."Game"
where "Game_ID"=tmp;

update "Game_Zone"."Credit_Card"
set "Balance"="Balance"-x,
      "Total_Points"="Total_Points"+new."Points_earned"
```

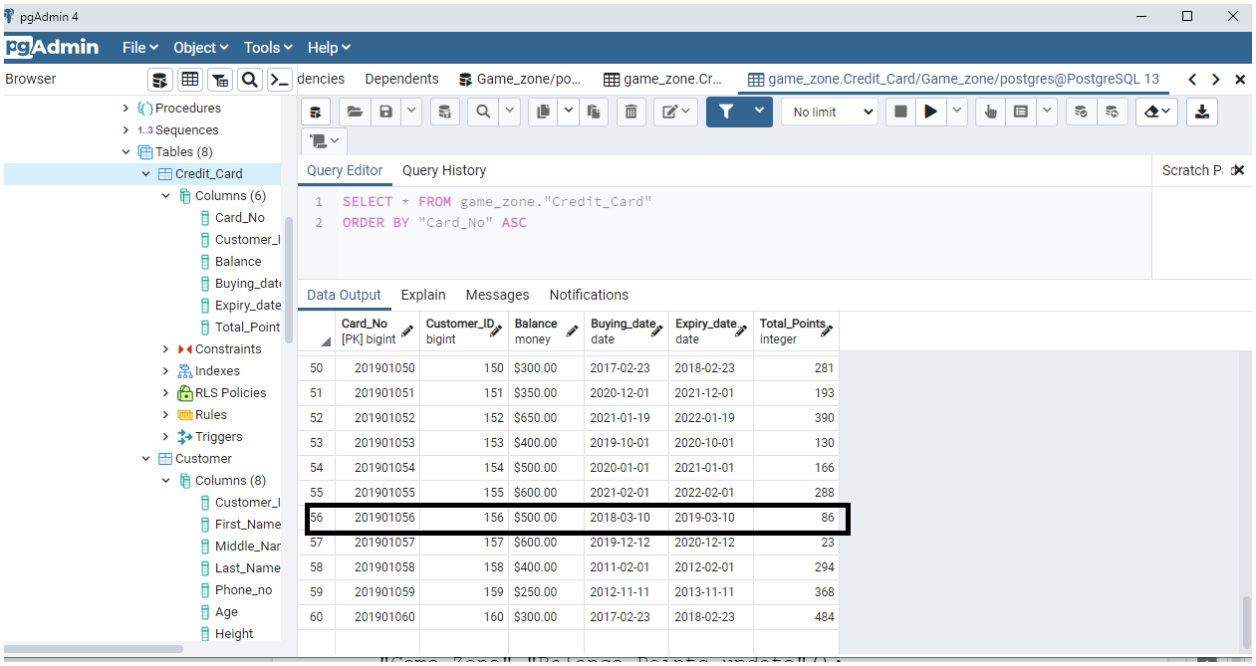
```
where "Customer_ID"=new."Customer_ID";

raise notice 'Triggered';
return NULL;
end
$BODY$;
```

Creating trigger in table:

```
CREATE TRIGGER "Balance_Points_update_trigger"
    AFTER INSERT
    ON "Game_Zone"."Game_History"
    FOR EACH ROW
    EXECUTE FUNCTION "Game_Zone"."Balance_Points_update"();
```



	Card_No [PK] bigint	Customer_ID bigint	Balance numeric	Buying_date date	Expiry_date date	Total_Points integer
48	201901048	148	600.00	2011-02-01	2012-02-01	339
49	201901049	149	250.00	2012-11-11	2013-11-11	314
50	201901050	150	300.00	2017-02-23	2018-02-23	281
51	201901051	151	350.00	2020-12-01	2021-12-01	193
52	201901052	152	650.00	2021-01-19	2022-01-19	390
53	201901053	153	400.00	2019-10-01	2020-10-01	130
54	201901054	154	500.00	2020-01-01	2021-01-01	166
55	201901055	155	600.00	2021-02-01	2022-02-01	288
56	201901056	156	700.00	2018-03-10	2019-03-10	30
57	201901057	157	600.00	2019-12-12	2020-12-12	23
58	201901058	158	400.00	2011-02-01	2012-02-01	294
59	201901059	159	250.00	2012-11-11	2013-11-11	368
60	201901060	160	300.00	2017-02-23	2018-02-23	484



Before inserting in game_history, a customer with id 156 has a balance of 700 and total_points 36. After inserting one tuple in game_history, that customer's balance reduced to 200 and total_points increased to 86 according to the inserted tuple.

3.Trigger for updating high score of game

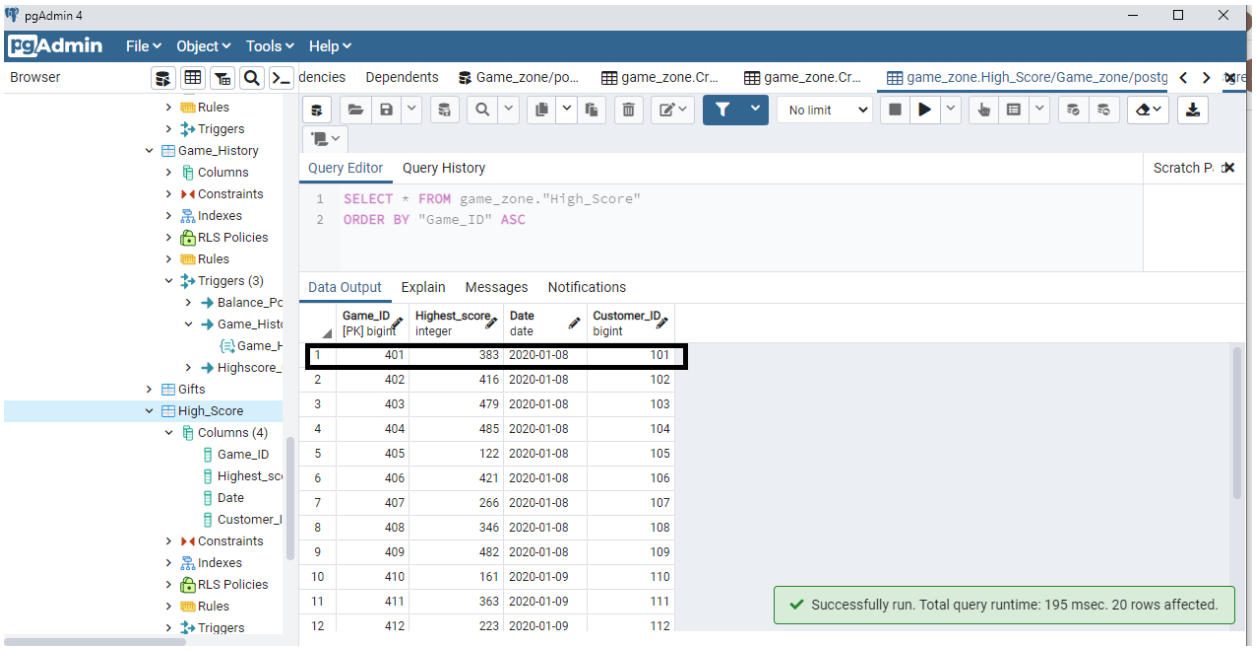
Trigger function SQL code:

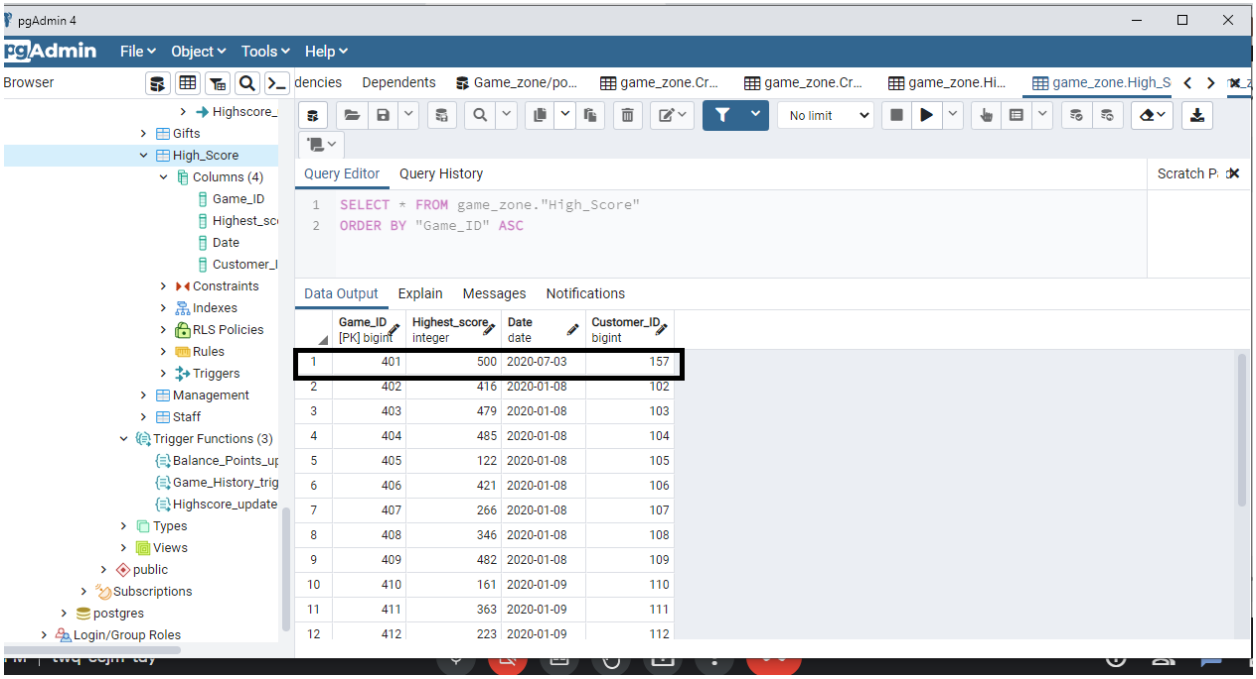
```
CREATE OR REPLACE FUNCTION "Game_Zone"."Highscore_update"()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    VOLATILE
    COST 100
AS $BODY$
declare
tmp bigint;
x integer;
y integer;
begin
tmp=new."Game_ID";
x=new."Points_earned";

select "Highest_score" into y from "Game_Zone"."High_Score"
where "Game_ID"=tmp;
```

```
if(x>y) then
       update "Game_Zone"."High_Score"
       set "Highest_score"=x,
             "Customer_ID"=new."Customer_ID",
             "Date"=date(new."Start_time")
       where "Game_ID"=tmp;
end if;

raise notice 'Triggered';
return NULL;
end
$BODY$;
```

Creating trigger in table:

```
CREATE TRIGGER "Highscore_update_trigger"
    AFTER INSERT
    ON "Game_Zone"."Game_History"
    FOR EACH ROW
    EXECUTE FUNCTION "Game_Zone"."Highscore_update"();
```





Before inserting in game_history, the high score of the game with id 401 is 383, scored by customer id 157 on 2020-01-08.

After inserting in game_history, the high score of the game with id 401 changed to 500, customer id changed to 157 and date changed to 2020-07-03.

4. Trigger for insertion in credit card table

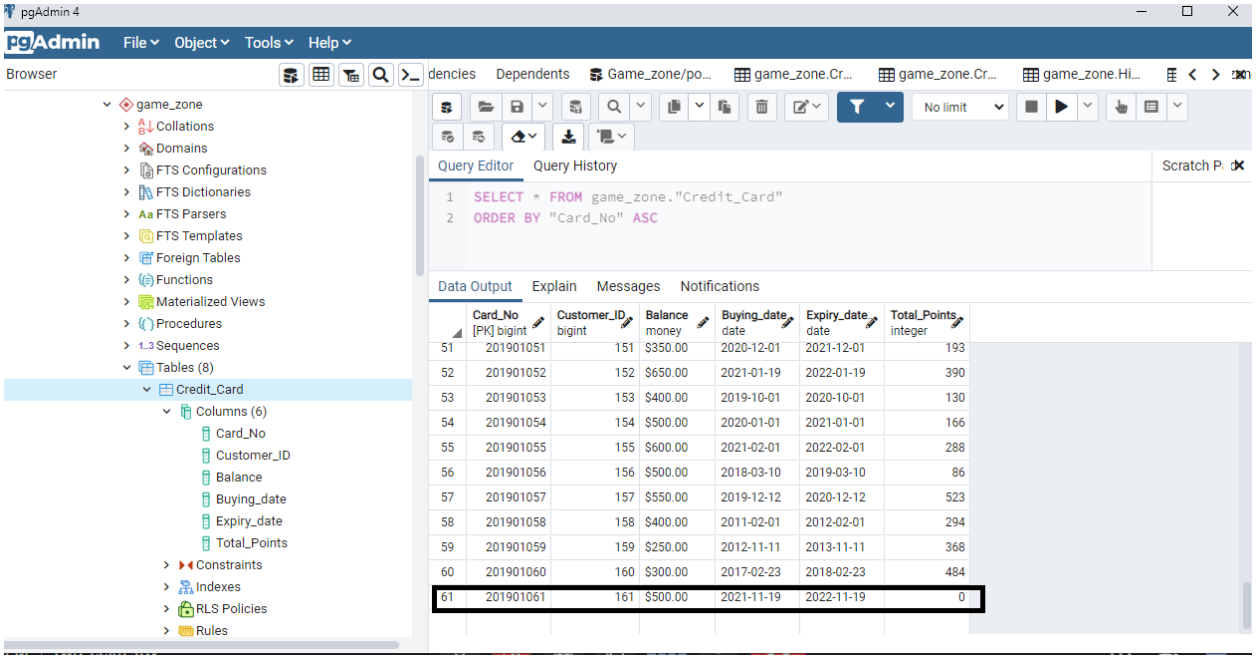
Trigger function SQL code:

```
CREATE FUNCTION "Game_Zone".credit_card_update()
    RETURNS trigger
    LANGUAGE 'plpgsql'
     NOT LEAKPROOF
AS $BODY$

declare
m bigint;

begin
select max("Card_No") into m from "game_zone"."Credit_Card";
insert into "game_zone"."Credit_Card"
values(m+1,new."Customer_ID",'$ 500',date(now()),date(now())+ interval '1
year',0);
raise notice 'Triggered';
return NULL;
```

```
end

$BODY$;

ALTER FUNCTION "Game_Zone".credit_card_update()
    OWNER TO postgres;
```

Creating trigger in table:

```
CREATE TRIGGER credit_card_update_trigger
    AFTER INSERT
    ON "Game_Zone"."Customer"
    FOR EACH ROW
    EXECUTE FUNCTION "Game_Zone".credit_card_update();
```



If a new customer is added into the customer table, the customer is given a new credit card. These details are added to the credit card table.

Function:

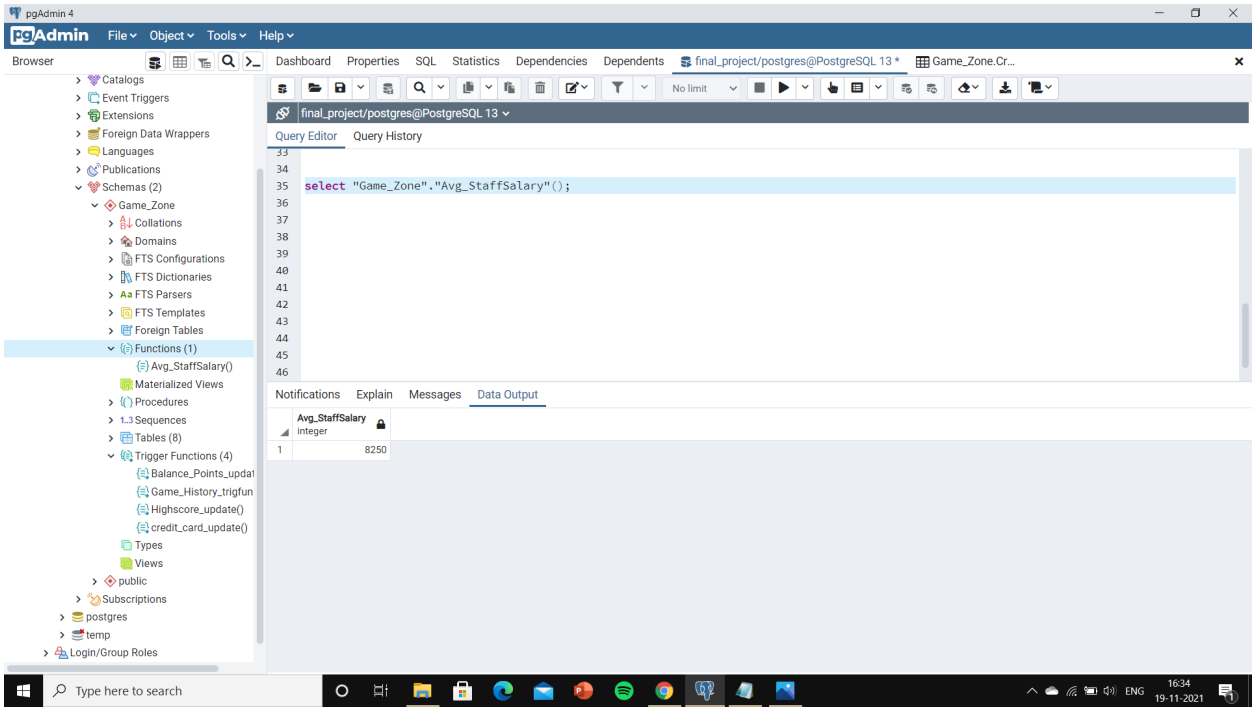
1. **To find average salary of staff**

```
CREATE FUNCTION "Game_Zone"."Avg_StaffSalary"()
    RETURNS integer
    LANGUAGE 'plpgsql'

AS $BODY$
declare
```

```
x integer;
begin
select avg("Salary") into x from "Game_Zone"."Staff";
return x;

end
$BODY$;

ALTER FUNCTION "Game_Zone"."Avg_StaffSalary"()
    OWNER TO postgres;
```

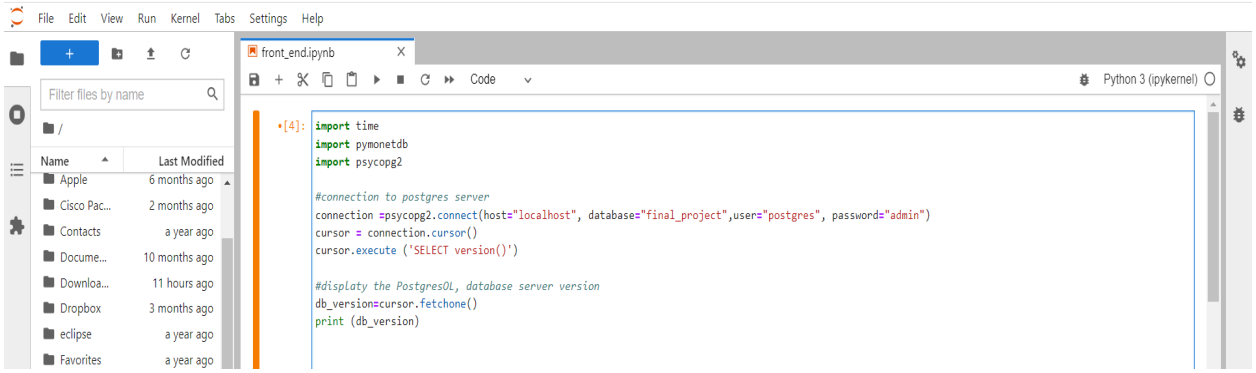
Call function:

```
select "Game_Zone"."Avg_StaffSalary"();
```



Section7: Project Code with output screenshots.

Code for connecting to postgres database:

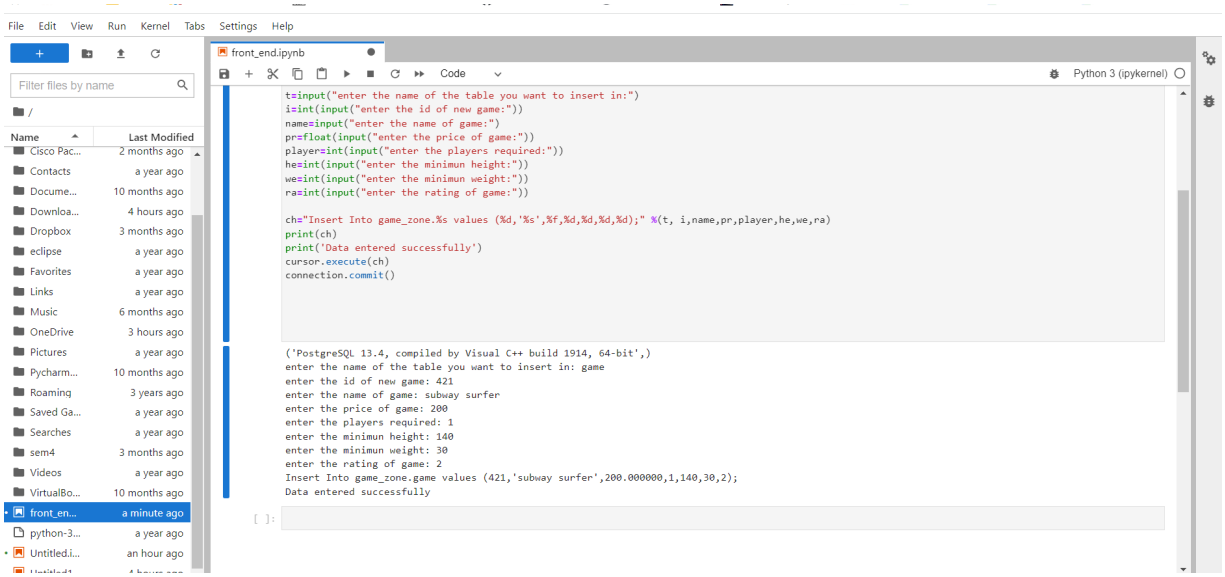


CODE:

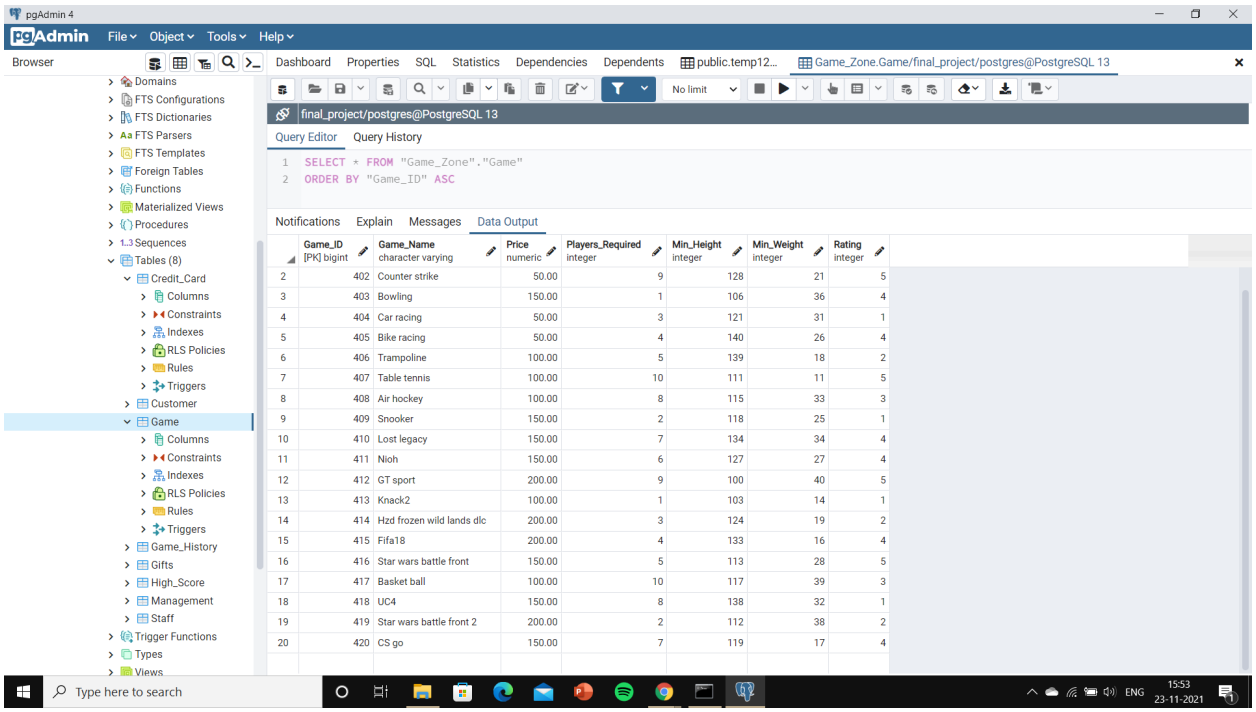
```
import time
import pymonetdb
import psycopg2

#connection to postgres server
connection =psycopg2.connect(host="localhost",
database="final_project",user="postgres", password="admin")
cursor = connection.cursor()
cursor.execute ('SELECT version()')

#display the PostgresOL, database server version
db_version=cursor.fetchone()
print (db_version)
```

```
#insertion in table of database
t=input("enter the name of the table you want to insert in:")
i=int(input("enter the id of new game:"))
name=input("enter the name of game:")
pr=float(input("enter the price of game:"))
player=int(input("enter the players required:"))
he=int(input("enter the minimun height:"))
we=int(input("enter the minimun weight:"))
ra=int(input("enter the rating of game:"))

ch="Insert Into game_zone.%s values (%d,'%s',%f,%d,%d,%d,%d);" %(t,
i,name,pr,player,he,we,ra)
print(ch)
print('Data entered successfully')
cursor.execute(ch)
connection.commit()


#query - 1
ch="select staff_id,salary from game_zone.staff"
cursor.execute (ch)
rows=cursor.fetchall()
print('staff_id   salary')
for r in rows:
    print('%d        %d ' %(r[0],r[1]))


#query -2
ch="select product_name from game_zone.gifts where available_stock > 10"
cursor.execute (ch)
rows=cursor.fetchall()
print('\nGift_name\n------\n')
for r in rows:
    print('%s ' %(r[0]))
```

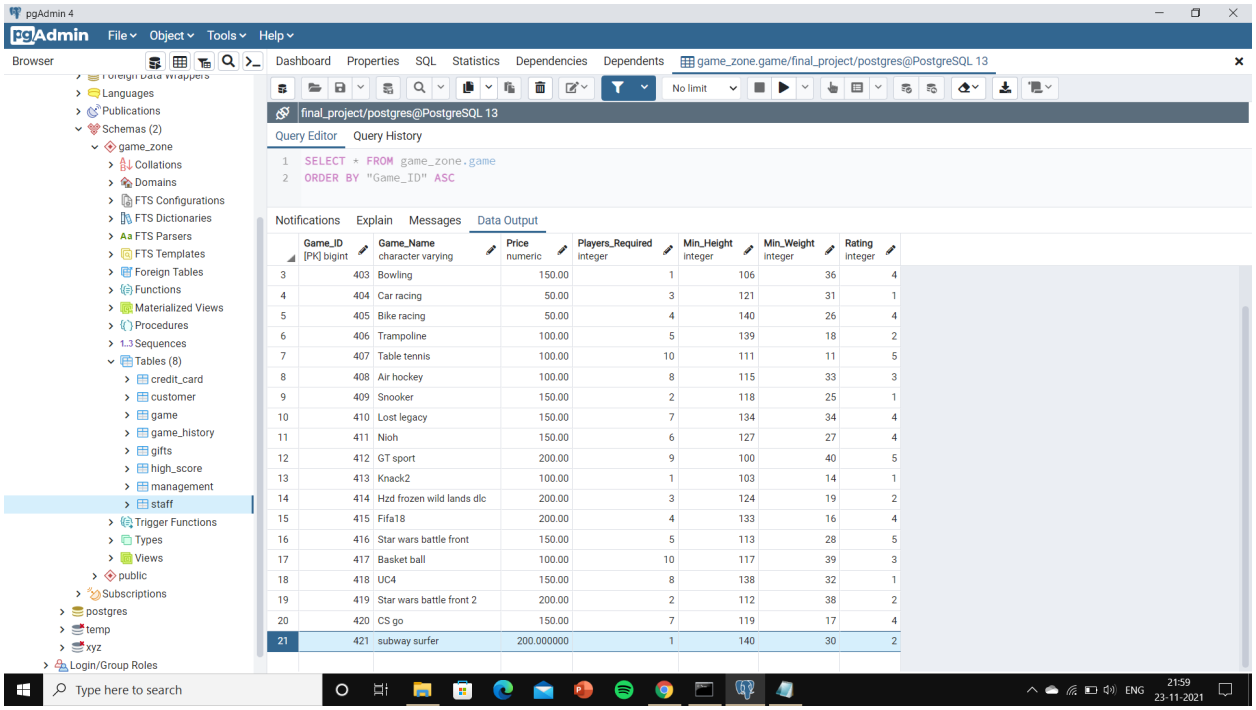
Insertion in table:



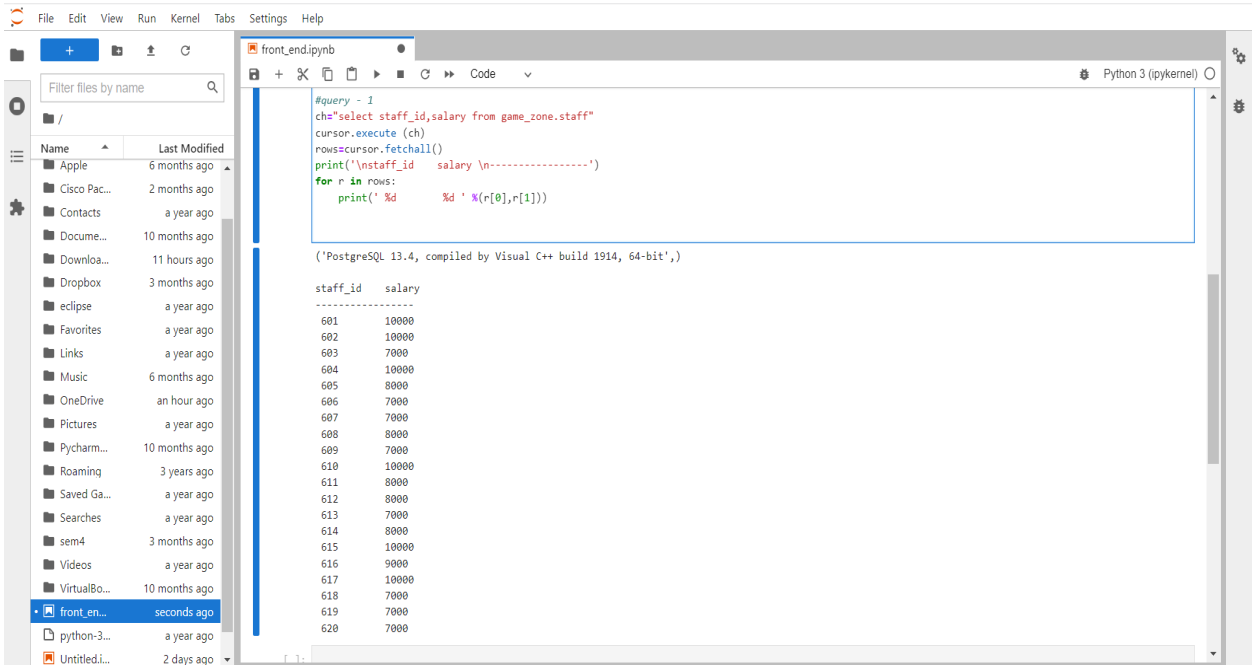
Before insertion there are only 20 games, with id of the last game 420. This is shown in the screenshot below.



After insertion of a new game is added with id 421, which can be seen from the following screenshot.



Query 1: Find staff id and it's salary.



Query 2: Find the gifts name whose available stock is greater than 10.