University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 10/15/2024

Group Number: 91

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jerry Qi	73382533	s6l7k	zqi07@students.cs.ubc.ca
Rapeewit Chanprakaisi	57529208	w2g6k	rchanpra@student.ubc.ca
Eric Xiang	90612029	g9e4y	ericxiang8@hotmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

Our project is a PC parts picker that assists users in building a custom PC. The domain of our project is custom PC building/computer hardware management/shopping assistant. The database will provide functionality that allows users to search, filter, and select computer hardware components for custom PC builds based on specific criteria, ensures component compatibility checks, and tracks price history for each product with real-time pricing and availability from various retailers.

The ER diagram you are basing your item #3 (below) on.

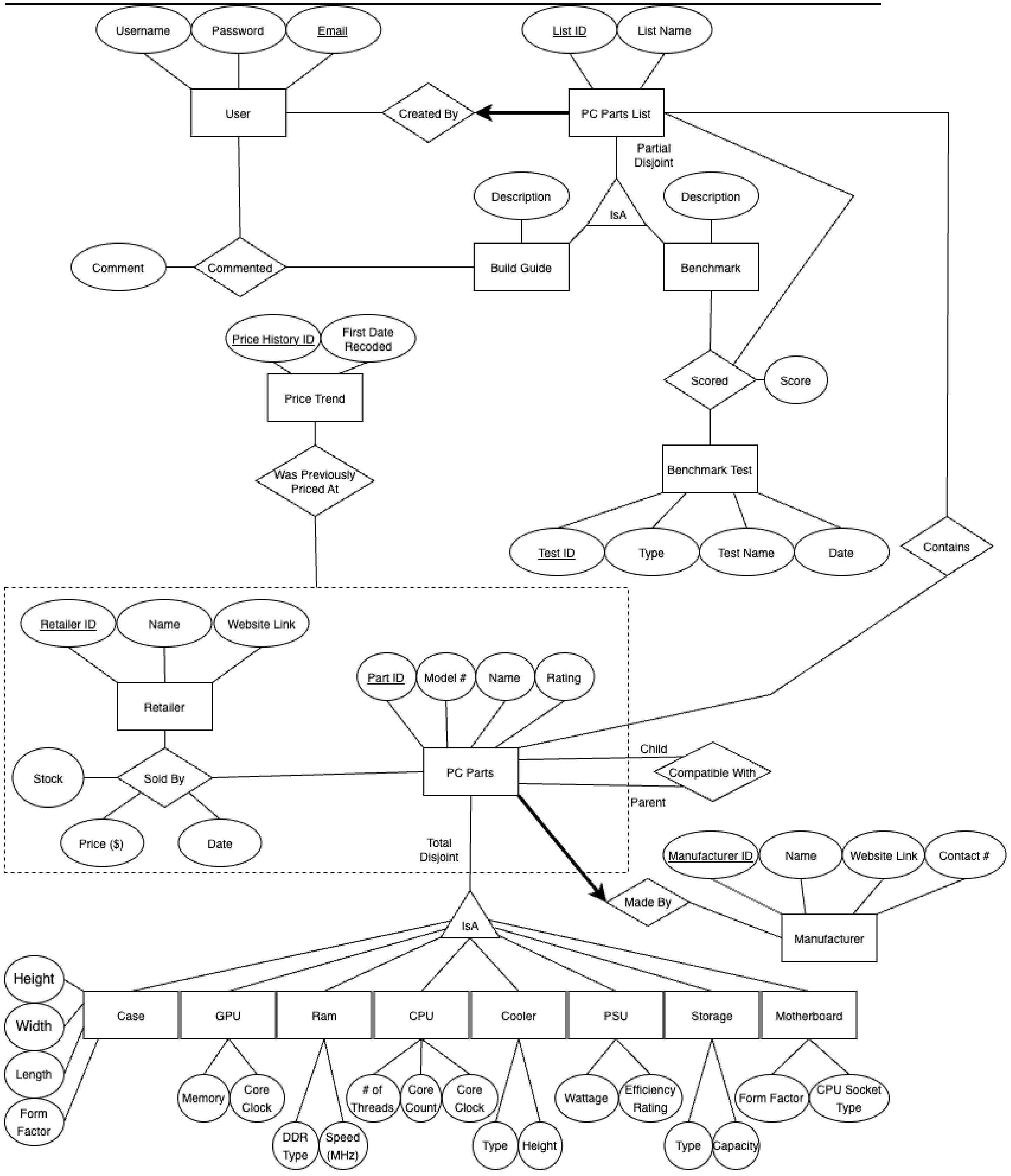
Changes made:

1. Changed the primary key of entity “Users” to be Email

2. Changed the PC Parts ISA to be more distinct in what makes each PC part unique from the rest. I.e added attributes to better allow filtering between each pc parts when looking them up in our database

3. Changed the box for aggregation on the “Sold By” relationship between Retailers and PC Parts, to better capture the relationship we are trying to show when using this aggregation for the “Previously Price At” relationship of Price Trend.

4. Also made the aggregation relationship be many to one, (one being price trends), many being the other relationship


Username
Password
Email
List ID
List Name
User
Created By
PC Parts List
Partial Disjoint
IsA
Description
Description
Comment
Commented
Build Guide
Benchmark
Price History ID
First Date Recoded
Price Trend
Scored
Score
Was Previously Priced At
Benchmark Test
Test ID
Type
Test Name
Date
Contains
Retailer ID
Name
Website Link
Retailer
Part ID
Model #
Name
Rating
Stock
Sold By
PC Parts
Child
Compatible With
Parent
Price ($)
Date
Total Disjoint
Manufacturer ID
Name
Website Link
Contact #
Made By
Manufacturer
IsA
Height
Width
Length
Form Factor
Case
GPU
Ram
CPU
Cooler
PSU
Storage
Motherboard
Memory
Core Clock
DDR Type
Speed (MHz)
# of Threads
Core Count
Core Clock
Type
Height
Wattage
Efficiency Rating
Type
Capacity
Form Factor
CPU Socket Type

University of British Columbia, Vancouver
Department of Computer Science

The schema derived from your ER diagram (above).
a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.
b. Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

Primary keys are underlined
Example: PartID means that "PartID" is the primary key for its table.

PCParts(
PartID: integer,
Name: char[20],
Model: char[20],
Rating: integer,
ManufacturerID: integer NOT NULL,
FK(ManufacturerID) References Manufacturer(ManufacturerID)
)
Case(
PartID: integer UNIQUE,
Height: integer,
Width: integer,
Length: integer,
FormFactor: char[20],
FK(PartID) References PCParts(PartID)
)
GPU(
PartID: integer UNIQUE,
Memory: integer,
CoreClock: integer,
FK(PartID) References PCParts(PartID)
)
Ram(
PartID: integer UNIQUE,
DDRType: char[20],
Speed: integer,
FK(PartID) References PCParts(PartID)
)
CPU(

PartID: integer UNIQUE,
ThreadCount: integer,
CoreCount: integer,
CoreClock: integer,
FK(PartID) References PCParts(PartID)
)
Cooler(
PartID: integer UNIQUE,
Type: char[20],
Height: integer,
FK(PartID) References PCParts(PartID)
)
PSU(
PartID: integer UNIQUE,
Wattage: integer,
EfficiencyRating: char[20],
FK(PartID) References PCParts(PartID)
)
Storage(
PartID: integer UNIQUE,
Type: char[20],
Capacity: integer,
FK(PartID) References PCParts(PartID)
)
Motherboard(
PartID: integer UNIQUE,
Formfactor: char[20],
Sockettype: char[20],
FK(PartID) References PCParts(PartID)
)
Retailer(
RetailerID: integer,
Name: char[20],
Website: char[50]
)
Manufacturer(
ManufacturerID: integer,
Name: char[20],

Website: char[50],
Contract: char[20]
)
Price Trend(
PricehistoryID: integer,
FirstDateRecorded: char[20]
RetailerID: integer
PartID: integer
        FK(PartID, RetailerID) References SoldBy(PartID, RetailerID),
)
User(
Email: char[50],
Username: char[20] UNIQUE,
Password: char[20]
ListID: integer NOT NULL,
FK(ListID) References PCPartsList(ListID)
)
PCPartsList(
ListID: integer,
ListName: char[50],
UserEmail: char[50],
FK(UserEmail) References User(Email)
)
Benchmark(
ListID: integer UNIQUE,
Description: char[1000],
FK(ListID) References PCPartsList(ListID)
)
BuildGuide(
ListID: integer UNIQUE,
Description: char[1000],
FK(ListID) References PCPartsList(ListID)
)
Benchmark-Test(
TestID: integer,
TestName: char[50],
Type: char[50],
Date: date

```
)
CompatibleWith(
        ChildPartID: integer,
        ParentPartID: integer,
        FK(ChildPartID) References PCParts(PartID),
        FK(ParentPartID) References PCParts(PartID)
)
SoldBy(
        RetailerID: integer,
        PartID: integer
        Stock: integer,
        Price: integer,
        Date: date,
        FK(PartID) References PCPartsPartID),
        FK(RetailerID) References Retailer(RetailerID)
)
Contains(
        ListID: integer,
        PartID: integer,
        FK(ListID) References PCPartsList(ListID),
        FK(PartID) References PCParts(PartID),
)
Scored(
        TestID: integer,
        ListID: integer,
        Score: integer,
        FK(TestID) References Benchmark-Test(TestID),
        FK(ListID) References Benchmark(ListID)
)
Commented(
Email: char[50],
ListID: integer,
Comment: char[1000],
FK(Email) References User(Email),
FK(ListID) References BuildGuide(ListID)
)
```

Functional Dependencies: Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key).

PC Parts:
Part ID → Name + Model + Rating + ManufacturerID
Name + Model → Rating

Case:
Part ID → Height + Width + Length + Form Factor

GPU:
Part ID → Memory + Core Clock

Ram:
Part ID → DDR Type + Speed

CPU:
Part ID → Threads Count + Core Count + Core Clock

Cooler:
Part ID → Type + Height

PSU:
Part ID → Wattage + Efficiency Rating

Storage:
Part ID → Type + Capacity

SoldBy:
Retailer ID + Part ID → Stock + Price + Date

Motherboard:
Part ID → Form Factor + CPU Socket Type

Retailer:
Retailer ID → Name + Website Link
Website Link → Name

Manufacturer:

Manufacturer ID → Name + Website Link + Contact
Website Link → Name + Contact

User:
Email → Username + Password
Username → Password

PC Parts List:
List ID → List Name + UserEmail

Benchmark:
List ID → Description

Build Guide:
List ID → Description

Price Trend:
Price History ID → First Date Recorded + RetailerID + PartID

Benchmark Test:
Test ID → Type + Test Name + Date

Normalization: Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization.
If the table only contains FD with PK then it is already normalized.

PCParts(
PartID: integer,
Name: char[20],
Model: char[20],
Rating: integer,
ManufacturerID: integer NOT NULL,
FK(ManufacturerID) References Manufacturer(ManufacturerID)
)

PC Parts:

Part ID → Name + Model + Rating + ManufacturerID
Name + Model → Rating

Decompose on Name + Model → Rating

P1(PartID, Name, Model, ManufacturerID)
P2(Name, Model, Rating)

```
Case(
PartID: integer UNIQUE,
Height: integer,
Width: integer,
Length: integer,
FormFactor: char[20],
FK(PartID) References PCParts(PartID)
)
GPU(
PartID: integer UNIQUE,
Memory: integer,
CoreClock: integer,
FK(PartID) References PCParts(PartID)
)
Ram(
PartID: integer UNIQUE,
DDRType: char[20],
Speed: integer,
FK(PartID) References PCParts(PartID)
)
CPU(
PartID: integer UNIQUE,
ThreadCount: integer,
CoreCount: integer,
CoreClock: integer,
FK(PartID) References PCParts(PartID)
)
Cooler(
PartID: integer UNIQUE,
Type: char[20],
```

Height: integer,
FK(PartID) References PCParts(PartID)
)
PSU(
PartID: integer UNIQUE,
Wattage: integer,
EfficiencyRating: char[20],
FK(PartID) References PCParts(PartID)
)
Storage(
PartID: integer UNIQUE,
Type: char[20],
Capacity: integer,
FK(PartID) References PCParts(PartID)
)
Motherboard(
PartID: integer UNIQUE,
Formfactor: char[20],
Sockettype: char[20],
FK(PartID) References PCParts(PartID)
)
Retailer(
RetailerID: integer,
Name: char[20],
Website: char[50]
)

Retailer:
Retailer ID → Name + Website Link
Website Link → Name
Decompose on Website Link → Name

R1(WebsiteLink, Name)
R2(RetailerID, Name, Website_Link)

Manufacturer(
ManufacturerID: integer,

Name: char[20],
Website: char[50],
Contract: char[20]
)

Manufacturer:
Manufacturer ID → Name + Website Link + Contact
Website Link → Name + Contact

Decompose on Website Link → Name + Contact
M1(ManufacturerID, Name, WebsiteLink, Contact) M2(WebsiteLink, Name, Contact)

Price Trend(
PricehistoryID: integer,
FirstDateRecorded: char[20]
RetailerID: integer
PartID: integer
        FK(PartID, RetailerID) References SoldBy(PartID, RetailerID),
)
User(
Email: char[50],
Username: char[20],
Password: char[20]
ListID: integer NOT NULL,
FK(ListID) References PCPartsList(ListID)
)

User:
Email → Username + Password
Username → Password

Decompose on Username → Password
U1(Email, Username, Password) U2(Username, Password)

PCPartsList(
ListID: integer,
ListName: char[50],

```
UserEmail:  char[50],
FK(UserEmail) References User(Email)
)
Benchmark(
ListID: integer UNIQUE,
Description: char[1000],
FK(ListID) References PCPartsList(ListID)
)
BuildGuide(
ListID: integer UNIQUE,
Description: char[1000],
FK(ListID) References PCPartsList(ListID)
)
Benchmark-Test(
TestID: integer,
TestName: char[50],
Type: char[50],
Date: date
)
CompatibleWith(
        ChildPartID: integer,
        ParentPartID: integer,
        FK(ChildPartID) References PCParts(PartID),
        FK(ParentPartID) References PCParts(PartID)
)
SoldBy(
        RetailerID: integer,
        PartID: integer
        Stock: integer,
        Price: integer,
        Date: date,
        FK(PartID) References PCPartsPartID),
        FK(RetailerID) References Retailer(RetailerID)
)
Contains(
        ListID: integer,
        PartID: integer,
        FK(ListID) References PCPartsList(ListID),
        FK(PartID) References PCParts(PartID),
Scored(
        TestID: integer,
        ListID: integer,
        Score: integer,
        FK(TestID) References Benchmark-Test(TestID),
        FK(ListID) References Benchmark(ListID)
)
Commented(
Email: char[50],
ListID: integer,
Comment: char[1000],
FK(Email) References User(Email),
FK(ListID) References BuildGuide(ListID)
)
```

The SQL DDL statements required to create all the tables from item #6.

```
CREATE TABLE PCParts (
  PartID INTEGER PRIMARY KEY,
  Name CHAR(20),
  Model CHAR(20),
  Rating INTEGER,
  ManufacturerID INTEGER NOT NULL,
  FOREIGN KEY (ManufacturerID) REFERENCES Manufacturer(ManufacturerID)
);

CREATE TABLE Case (
  PartID INTEGER PRIMARY KEY,
  Height INTEGER,
  Width INTEGER,
  Length INTEGER,
  FormFactor CHAR(20),
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE GPU (
  PartID INTEGER PRIMARY KEY,
  Memory INTEGER,
  CoreClock INTEGER,
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE Ram (
  PartID INTEGER PRIMARY KEY,
  DDRType CHAR(20),
  Speed INTEGER,
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE CPU (
  PartID INTEGER PRIMARY KEY,
  ThreadCount INTEGER,
  CoreCount INTEGER,
  CoreClock INTEGER,
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE Cooler (
  PartID INTEGER PRIMARY KEY,
  Type CHAR(20),
  Height INTEGER,
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE PSU (
  PartID INTEGER PRIMARY KEY,
  Wattage INTEGER,
  EfficiencyRating CHAR(20),
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE Storage (
  PartID INTEGER PRIMARY KEY,
  Type CHAR(20),
  Capacity INTEGER,
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE Motherboard (
  PartID INTEGER PRIMARY KEY,
  FormFactor CHAR(20),
  SocketType CHAR(20),
  FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE Retailer (
  RetailerID INTEGER PRIMARY KEY,
  Name CHAR(20),
  Website CHAR(50)
);

CREATE TABLE Manufacturer (
  ManufacturerID INTEGER PRIMARY KEY,
  Name CHAR(20),
  Website CHAR(50),
  Contract CHAR(20)
);

CREATE TABLE PriceTrend (
  PriceHistoryID INTEGER PRIMARY KEY,
  FirstDateRecorded CHAR(20),
  RetailerID INTEGER,
  PartID INTEGER,
  FOREIGN KEY (PartID, RetailerID) REFERENCES SoldBy(PartID, RetailerID)
);

CREATE TABLE User (
  Email CHAR(50) PRIMARY KEY,
  Username CHAR(20) UNIQUE,
  Password CHAR(20),
  ListID INTEGER NOT NULL,
  FOREIGN KEY (ListID) REFERENCES PCPartsList(ListID)
```

```
);


CREATE TABLE PCPartsList (
   ListID INTEGER PRIMARY KEY,
   ListName CHAR(50),
   UserEmail CHAR(50),
   FOREIGN KEY (UserEmail) REFERENCES User(Email)
);

CREATE TABLE Benchmark (
   ListID INTEGER PRIMARY KEY UNIQUE,
   Description CHAR(1000),
   FOREIGN KEY (ListID) REFERENCES PCPartsList(ListID)
);

CREATE TABLE BuildGuide (
   ListID INTEGER PRIMARY KEY UNIQUE,
   Description CHAR(1000),
   FOREIGN KEY (ListID) REFERENCES PCPartsList(ListID)
);

CREATE TABLE BenchmarkTest (
   TestID INTEGER PRIMARY KEY,
   TestName CHAR(50),
   Type CHAR(50),
   Date DATE
);

CREATE TABLE CompatibleWith (
   ChildPartID INTEGER,
   ParentPartID INTEGER,
   PRIMARY KEY (ChildPartID, ParentPartID),
   FOREIGN KEY (ChildPartID) REFERENCES PCParts(PartID),
   FOREIGN KEY (ParentPartID) REFERENCES PCParts(PartID)
);

CREATE TABLE SoldBy (
   RetailerID INTEGER,
   PartID INTEGER,
   Stock INTEGER,
   Price INTEGER,
   Date DATE,
   PRIMARY KEY (RetailerID, PartID),
   FOREIGN KEY (RetailerID) REFERENCES Retailer(RetailerID),
   FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE Contains (
   ListID INTEGER,
   PartID INTEGER,
   PRIMARY KEY (ListID, PartID),
   FOREIGN KEY (ListID) REFERENCES PCPartsList(ListID),
   FOREIGN KEY (PartID) REFERENCES PCParts(PartID)
);

CREATE TABLE Scored (
   TestID INTEGER,
   ListID INTEGER,
   Score INTEGER,
   PRIMARY KEY (TestID, ListID),
   FOREIGN KEY (TestID) REFERENCES BenchmarkTest(TestID),
   FOREIGN KEY (ListID) REFERENCES Benchmark(ListID)
);

CREATE TABLE Commented (
   Email CHAR(50),
   ListID INTEGER,
   Comment CHAR(1000),
   PRIMARY KEY (Email, ListID),
   FOREIGN KEY (Email) REFERENCES User(Email),
   FOREIGN KEY (ListID) REFERENCES BuildGuide(ListID)
);
```

INSERT statements to populate each table with at least 5 tuples.

```
INSERT INTO PCParts (PartID, Name, Model, Rating, ManufacturerID)
VALUES
(1, 'RTX 3080', 'Founders', 9, 3),
(2, 'Ryzen 5 5600X', 'Vermeer', 8, 2),
(3, 'Hyper 212', 'EVO', 7, 4),
(4, '850W PSU', 'RM850x', 9, 4),
(5, '1TB SSD', '870 EVO', 10, 5);

INSERT INTO Case (PartID, Height, Width, Length, FormFactor)
VALUES
(6, 450, 210, 460, 'ATX'),
(7, 400, 200, 450, 'Micro-ATX'),
(8, 370, 180, 420, 'Mini-ITX'),
(9, 480, 220, 490, 'ATX'),
(10, 350, 170, 410, 'Mini-ITX');


INSERT INTO GPU (PartID, Memory, CoreClock)
VALUES
(1, 10, 1440),
(11, 8, 1600),
(12, 24, 1800),
(13, 6, 1400),
(14, 16, 2000);

INSERT INTO Ram (PartID, DDRType, Speed)
VALUES
(15, 'DDR4', 3200),
(16, 'DDR5', 4800),
(17, 'DDR4', 3600),
(18, 'DDR3', 1600),
(19, 'DDR5', 5200);

INSERT INTO CPU (PartID, ThreadCount, CoreCount, CoreClock)
VALUES
(2, 12, 6, 3800),
(20, 24, 12, 4000),
(21, 8, 4, 3000),
(22, 16, 8, 3600),
(23, 32, 16, 4600);

INSERT INTO Cooler (PartID, Type, Height)
VALUES
(3, 'Air', 158),
(24, 'Liquid', 240),
(25, 'Air', 140),
(26, 'Liquid', 280),
(27, 'Air', 160);

INSERT INTO PSU (PartID, Wattage, EfficiencyRating)
VALUES
(4, 850, 'Gold'),
(28, 750, 'Platinum'),
(29, 1000, 'Gold'),
(30, 600, 'Bronze'),
(31, 1200, 'Platinum');

INSERT INTO Storage (PartID, Type, Capacity)
VALUES
(5, 'SSD', 1024),
(32, 'HDD', 2000),
(33, 'NVMe', 512),
(34, 'SSD', 256),
(35, 'HDD', 1000);

INSERT INTO Motherboard (PartID, FormFactor, SocketType)
VALUES
(36, 'ATX', 'AM4'),
(37, 'Micro-ATX', 'LGA1200'),
(38, 'Mini-ITX', 'AM5'),
(39, 'ATX', 'LGA1700'),
(40, 'Micro-ATX', 'AM4');

INSERT INTO Retailer (RetailerID, Name, Website)
VALUES
```

```
(1, 'Amazon', 'www.amazon.com'),
(2, 'Best Buy', 'www.bestbuy.com'),
(3, 'Newegg', 'www.newegg.com'),
(4, 'Micro Center', 'www.microcenter.com'),
(5, 'B&H', 'www.bhphotovideo.com');

INSERT INTO Manufacturer (ManufacturerID, Name, Website, Contract)
VALUES
(1, 'Intel', 'www.intel.com', 'Contract A'),
(2, 'AMD', 'www.amd.com', 'Contract B'),
(3, 'NVIDIA', 'www.nvidia.com', 'Contract C'),
(4, 'Corsair', 'www.corsair.com', 'Contract D'),
(5, 'Samsung', 'www.samsung.com', 'Contract E');

INSERT INTO PriceTrend(PricehistoryID, FirstDateRecorded, RetailerID, PartID)
VALUES
(1, ‘2021-02-01’, '2', '6'),
(2, ‘2022-01-11’, '4', '1'),
(3, ‘2023-05-01’, '1', '3'),
(4, ‘2024-01-05’, '3', '5'),
(5, ‘2024-06-01’, '5', '4');

INSERT INTO User (Email, Username, Password, ListID)
VALUES
('alice@gmail.com', 'alice', 'pass1234', 1),
('bob@gmail.com', 'bobster', 'password', 2),
('eve@gmail.com', 'evee', 'hunter2', 3),
('mallory@gmail.com', 'mal', 'qwerty', 4),
('trent@gmail.com', 'trent22', 'asdf1234', 5);

INSERT INTO PCPartsList (ListID, ListName, UserEmail)
VALUES
(1, 'Gaming Build', 'alice@gmail.com'),
(2, 'Workstation Build', 'bob@gmail.com'),
(3, 'Streaming Build', 'eve@gmail.com'),
(4, 'Budget Build', 'mallory@gmail.com'),
(5, 'High-End Build', 'trent@gmail.com');
```

```
INSERT INTO Benchmark (ListID, Description)
VALUES
(1, 'Gaming benchmark using 3DMark and Cyberpunk 2077.'),
(2, 'Workstation build tested with Blender rendering.'),
(3, 'Streaming setup tested with OBS and Twitch.'),
(4, 'Budget gaming performance tested with indie titles.'),
(5, 'High-end build used for VR and AI workloads.');

INSERT INTO BuildGuide (ListID, Description)
VALUES
(1, 'Guide for building a gaming PC.'),
(2, 'Instructions for assembling a workstation.'),
(3, 'Steps for setting up a streaming PC.'),
(4, 'Affordable build guide for casual gamers.'),
(5, 'Advanced guide for VR and AI setups.');



INSERT INTO Benchmark_Test (TestID, TestName, Type, Date)
VALUES
(1, 'Cinebench R23', 'CPU', '2024-01-01'),
(2, '3DMark', 'GPU', '2024-01-02'),
(3, 'Prime95', 'CPU', '2024-01-03'),
(4, 'CrystalDiskMark', 'Storage', '2024-01-04'),
(5, 'UserBenchmark', 'Overall', '2024-01-05');

INSERT INTO CompatibleWith (ChildPartID, ParentPartID)
VALUES
(1, 2),
(2, 3),
(3, 4),
(4, 5),
(5, 1);

INSERT INTO SoldBy (RetailerID, PartID, Stock, Price, Date)
VALUES
(1, 1, 100, 699, '2023-10-01'),
(2, 2, 50, 299, '2023-10-02'),
(3, 3, 75, 49, '2023-10-03'),
(4, 4, 30, 129, '2023-10-04'),
(5, 5, 200, 99, '2023-10-05');

INSERT INTO Contains (ListID, PartID)
VALUES
(1, 2),
(2, 3),
(3, 4),
(4, 5),
(5, 1);

INSERT INTO Scored (TestID, ListID, Score)
VALUES
(1, 2, 99),
(2, 3, 54),
(3, 4, 86),
(4, 5, 18),
(5, 1, 88);

INSERT INTO Commented (Email, ListID, Comment)
VALUES
('alice@gmail.com’, 2, ‘awesome build’),
('bob@gmail.com’, 3, ‘smh this build is not it’),
('eve@gmail.com’, 4, ‘i disagree on your CPU selection’),
('mallory@gmail.com’, 5, insane build’),
('trent@gmail.com’, 1, ‘this build is very questionable’);
```