Username: wenqingx Student ID: 813044

1. Write a series of statements to create the missing tables

```
-- -----------------------------------------------------
-- Table `Consumer`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Consumer` ;
CREATE TABLE IF NOT EXISTS `Consumer` (
  `UserName` VARCHAR(50) NOT NULL,
  `FirstName` VARCHAR(50) NULL,
  `LastName` VARCHAR(50) NULL,
  `email` VARCHAR(100) NULL,
  PRIMARY KEY (`UserName`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Licence`
-- SET FOREIGN_KEY_CHECKS = 0;
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Licence` ;
CREATE TABLE IF NOT EXISTS `Licence` (
  `SoftwareId` INT NOT NULL,
  `UserName` VARCHAR(50) NOT NULL,
  `PurchaseTime` DATETIME NOT NULL,
  `Installed` BOOLEAN NOT NULL,
  PRIMARY KEY (`SoftwareId`, `UserName`, `PurchaseTime`),
    FOREIGN KEY (`SoftwareId`)
    REFERENCES `Software` (`SoftwareId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
    FOREIGN KEY (`UserName`)
    REFERENCES `Consumer` (`UserName`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Location`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Location` ;
CREATE TABLE IF NOT EXISTS `Location` (
  `idLocation` INT NOT NULL AUTO_INCREMENT,
  `StreetNumber` SMALLINT NULL,
  `StreetNumberSuffix` VARCHAR(20) NULL,
  `StreetName` VARCHAR(50) NULL,
  `StreetType` VARCHAR(20) NULL,
  `MinorMunicipality` VARCHAR(50) NULL,
  `MajorMunicipality` VARCHAR(50) NULL,
  `GoverningDistrict` VARCHAR(50) NULL,
  `PostalArea` VARCHAR(4) NULL,
  `Country` VARCHAR(50) NULL,
  PRIMARY KEY (`idLocation`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `ConsumerLocation`
-- SET FOREIGN_KEY_CHECKS = 0;
-- -----------------------------------------------------
DROP TABLE IF EXISTS `ConsumerLocation` ;
CREATE TABLE IF NOT EXISTS `ConsumerLocation` (
  `UserName` VARCHAR(50) NOT NULL,
  `idLocation` INT NOT NULL,
  `ValidFrom` DATE NOT NULL,
  `ValidTo` DATE NULL,
  PRIMARY KEY (`UserName`, `idLocation`, `ValidFrom`),
    FOREIGN KEY (`UserName`)
    REFERENCES `Consumer` (`UserName`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
    FOREIGN KEY (`idLocation`)
    REFERENCES `Location` (`idLocation`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

2. Write a series of SQL statements to populate the database for 5 consumers

```
INSERT INTO Consumer VALUES ("wenqingx", "Wenqing", "Xue",
"wenqingx@student.unimelb.edu.au");
INSERT INTO Consumer VALUES ("brucew", "Bruce", "Wayne",
"iambateman@wayne.com");
INSERT INTO Consumer VALUES ("sherlockh", "Sherlock", "Holmes", NULL);
INSERT INTO Consumer VALUES ("jamesb", "James", "Bond", "007@mi6.com.uk");
INSERT INTO Consumer VALUES ("harryp", "Harry", "Potter",
"harryp@student.hogwarts.edu.uk");

INSERT INTO Location VALUES (DEFAULT, NULL, NULL, NULL, NULL, NULL, "Gotham
City", NULL, NULL, "United States of America");
INSERT INTO Location VALUES (DEFAULT, 4, NULL, "Privet", "Street", NULL, "Little
Whinging", "Surrey", NULL, "United Kingdom");
INSERT INTO Location VALUES (DEFAULT, 61, NULL, "Horseferry", "Road", NULL,
"Westminster", "London", "SW1", "United Kingdom");
INSERT INTO Location VALUES (DEFAULT, 221, "B", "Baker", "Street", NULL,
"Marylebone", "London", "NW1", "United Kingdom");
```

Username: wenqingx   Student ID: 813044

```
INSERT INTO Location VALUES (DEFAULT, 757, NULL, "Swanston", "Street", NULL,
"Parkville", "Victoria", "3052", "Australia");

INSERT INTO ConsumerLocation VALUES ("brucew", 1, "1939-05-27", NULL);
INSERT INTO ConsumerLocation VALUES ("harryp", 2, "1990-07-31", NULL);
INSERT INTO ConsumerLocation VALUES ("jamesb", 3, "1968-04-13", NULL);
INSERT INTO ConsumerLocation VALUES ("sherlockh", 4, "1881-01-01", NULL);
INSERT INTO ConsumerLocation VALUES ("wenqingx", 5, "2016-02-25", NULL);

INSERT INTO Platform VALUES (1, "PlayStation 3");
INSERT INTO Platform VALUES (2, "PlayStation 4");
INSERT INTO Platform VALUES (3, "PlayStation Portable");
INSERT INTO Platform VALUES (4, "PlayStation Vita");
INSERT INTO Platform VALUES (5, "Nintendo 3DS");
INSERT INTO Platform VALUES (6, "Nintendo Wii U");
INSERT INTO Platform VALUES (7, "OS X");
INSERT INTO Platform VALUES (8, "iOS");
INSERT INTO Platform VALUES (9, "Windows");
INSERT INTO Platform VALUES (10, "Android");

INSERT INTO Software VALUES (1, "Batman: Arkham Asylum", 2, 29.95, 0.95, 1,
"action-adventure video game", 2009, "http://rocksteadyltd.com/#arkham-asylum");
INSERT INTO Software VALUES (2, "Batman: Arkham City", 1, 49.95, 0.95, 1, "action-
adventure video game", 2011, "http://rocksteadyltd.com/#arkham-city");
INSERT INTO Software VALUES (3, "Batman: Arkham Knight", 1, 89.95, 0.95, 2,
"action-adventure video game", 2015, "http://rocksteadyltd.com/#arkham-knight");
INSERT INTO Software VALUES (4, "Harry Potter Spells", 1, 1.95, 0, 8, "action game",
2012, "https://www.warnerbros.co.uk/games/harry-potter-spells");
INSERT INTO Software VALUES (5, "Grand Theft Auto V", 3, 89.95, 0, 9, "action-
adventure game", 2013, "http://www.rockstargames.com/V/");
INSERT INTO Software VALUES (6, "Counter-Strike: Global Offensive", 2, 89.95, 0, 9,
"first-person shooter game", 2012, "www.counter-strike.net");
INSERT INTO Software VALUES (7, "Sherlock Holmes: The Devil's Daughter", 1, 59.95,
0.95, 2, "adventure game", 2016, "http://sherlockholmes-games.com");
INSERT INTO Software VALUES (8, "Overwatch", 2, 89.95, 0, 9, "first-person shooter
game", 2016, "https://playoverwatch.com");

INSERT INTO Licence VALUES (1, "brucew", "2009-08-25 09:00:01", TRUE);
INSERT INTO Licence VALUES (2, "brucew", "2011-10-18 10:59:20", TRUE);
INSERT INTO Licence VALUES (3, "brucew", "2015-06-23 23:59:59", FALSE);
INSERT INTO Licence VALUES (4, "harryp", "2014-06-04 21:15:14", TRUE);
INSERT INTO Licence VALUES (5, "jamesb", "2013-09-17 11:11:11", TRUE);
INSERT INTO Licence VALUES (5, "wenqingx", "2014-12-25 14:50:21", FALSE);
INSERT INTO Licence VALUES (6, "jamesb", "2012-08-21 22:45:03", TRUE);
INSERT INTO Licence VALUES (7, "sherlockh", "2016-06-10 09:32:51", FALSE);
INSERT INTO Licence VALUES (8, "wenqingx", "2016-07-15 00:00:01", TRUE);
```

3. Write a series of SQL queries to appropriately change the location

```
-- ---------------------------------------------------
-- In case in safe update mode
-- SET SQL_SAFE_UPDATES = 0;
-- ---------------------------------------------------
UPDATE ConsumerLocation
SET ValidTo = CURDATE()
WHERE UserName = "wenqingx"
AND ValidTo IS NULL;

INSERT INTO Location
VALUES (DEFAULT, 123, NULL, "Pake", "Street", NULL, "Parkville", "Melbourne",
"9999", NULL);

INSERT INTO ConsumerLocation
VALUES ("wenqingx", LAST_INSERT_ID(), CURDATE(), NULL);
```

4. Count iOS apps having a distribution cost which is less than 20% of price

```
SELECT COUNT(DISTINCT SoftwareId) AS Count_iOS
FROM Software INNER JOIN Platform
ON Software.idPlatform = Platform.idPlatform
WHERE Platform.Name = "iOS"
AND Software.DistributionCost < Software.Price * 0.2
ORDER BY Count_iOS;
```

5. Count iOS apps which were released during each past decade

```
SELECT CONCAT((YearOfRelease DIV 10)*10, "s") AS Decade,
COUNT(DISTINCT SoftwareId) AS Count_iOS
FROM Software INNER JOIN Platform
ON Platform.idPlatform = Software.idPlatform
WHERE Platform.Name = "iOS"
AND YearOfRelease < 2010
GROUP BY Decade
ORDER BY Decade;
```

6. List Software Developers and count of iOS apps they created

```
SELECT idStaff, FirstName, LastName, COUNT(DISTINCT SoftwareId) AS Count_iOS
FROM Development NATURAL JOIN Software
NATURAL JOIN JobTitle NATURAL JOIN Staff
INNER JOIN Platform ON Platform.idPlatform = Software.idPlatform
WHERE OfficialJobTitle = "Software Developer"
AND Platform.Name = "iOS"
GROUP BY idStaff
ORDER BY idStaff;
```

7. List who has written code for the most and least software, including count of development projects

```
SELECT CONCAT_WS(" ", FirstName, LastName) AS Staff_Name, Count_APP FROM
(SELECT idStaff, FirstName, LastName, Count(SoftwareId) AS Count_APP
FROM Development NATURAL JOIN Staff NATURAL JOIN JobTitle
WHERE OfficialJobTitle = "Software Developer"
GROUP BY idStaff) AS Table_1 NATURAL JOIN
(SELECT MAX(Count_APP) AS Count_MAX, MIN(Count_APP) AS Count_MIN FROM
(SELECT idStaff, FirstName, LastName, Count(SoftwareId) AS Count_APP
FROM Development NATURAL JOIN Staff NATURAL JOIN JobTitle
WHERE OfficialJobTitle = "Software Developer"
GROUP BY idStaff) AS Table_1) AS Table_2
WHERE Count_APP = Count_MAX OR Count_APP = Count_MIN
ORDER BY Count_APP DESC;
```

8. List the 10 most purchased software starting with “i” or “e”

```
SELECT Table_1.SoftwareId, Table_1.Name, Table_1.Price,
Table_1.CurrentVersion, Count_APP, idStaff FROM
(SELECT DISTINCT SoftwareId, Name, COUNT(UserName) AS Count_APP
FROM Licence NATURAL JOIN Software
WHERE BINARY Name LIKE "i%" OR Name LIKE "e%"
GROUP BY SoftwareId
ORDER BY Count_APP DESC
LIMIT 10) AS Table_1 NATURAL JOIN Development NATURAL JOIN Staff
ORDER BY Count_APP DESC
```

9. List the software has not been purchased, shown by domain name

```
SELECT DISTINCT SoftwareId, Name,
SUBSTRING_INDEX(SUBSTRING_INDEX(Website, "://", -1), "/", 1) AS Domain_Name
FROM Software
WHERE SoftwareId NOT IN
(SELECT SoftwareId FROM Licence)
ORDER BY SoftwareId;
```

10. List the different version 1 software titles are installed for consumers in Australia

```
SELECT COUNT(Name) AS Count_APP
FROM Software NATURAL JOIN Licence
NATURAL JOIN ConsumerLocation
Natural JOIN Location
WHERE CurrentVersion = 1
AND Installed = TRUE
AND Country = "Australia"
ORDER BY Count_APP;
```

11. List the top 10 locations of paying consumer currently live, including full location and number of occupants

```
SELECT CONCAT_WS(", ", StreetNumber, StreetNumberSuffix,
StreetName, StreetType, MinorMunicipality, MajorMunicipality,
GoverningDistrict, PostalArea, Country) AS Address,
COUNT(UserName IN (SELECT UserName FROM ConsumerLocation)) AS Occupant
FROM Location NATURAL JOIN ConsumerLocation
WHERE ValidTo IS NULL
AND UserName IN (SELECT UserName FROM Licence)
GROUP BY idLocation
ORDER BY Occupant DESC
LIMIT 10;
```

12. List the consumers have never had a location recorded in database

```
SELECT UserName, FirstName, LastName FROM Consumer
WHERE UserName NOT IN
(SELECT UserName FROM ConsumerLocation)
ORDER BY UserName;
```