Car Rental DBMS

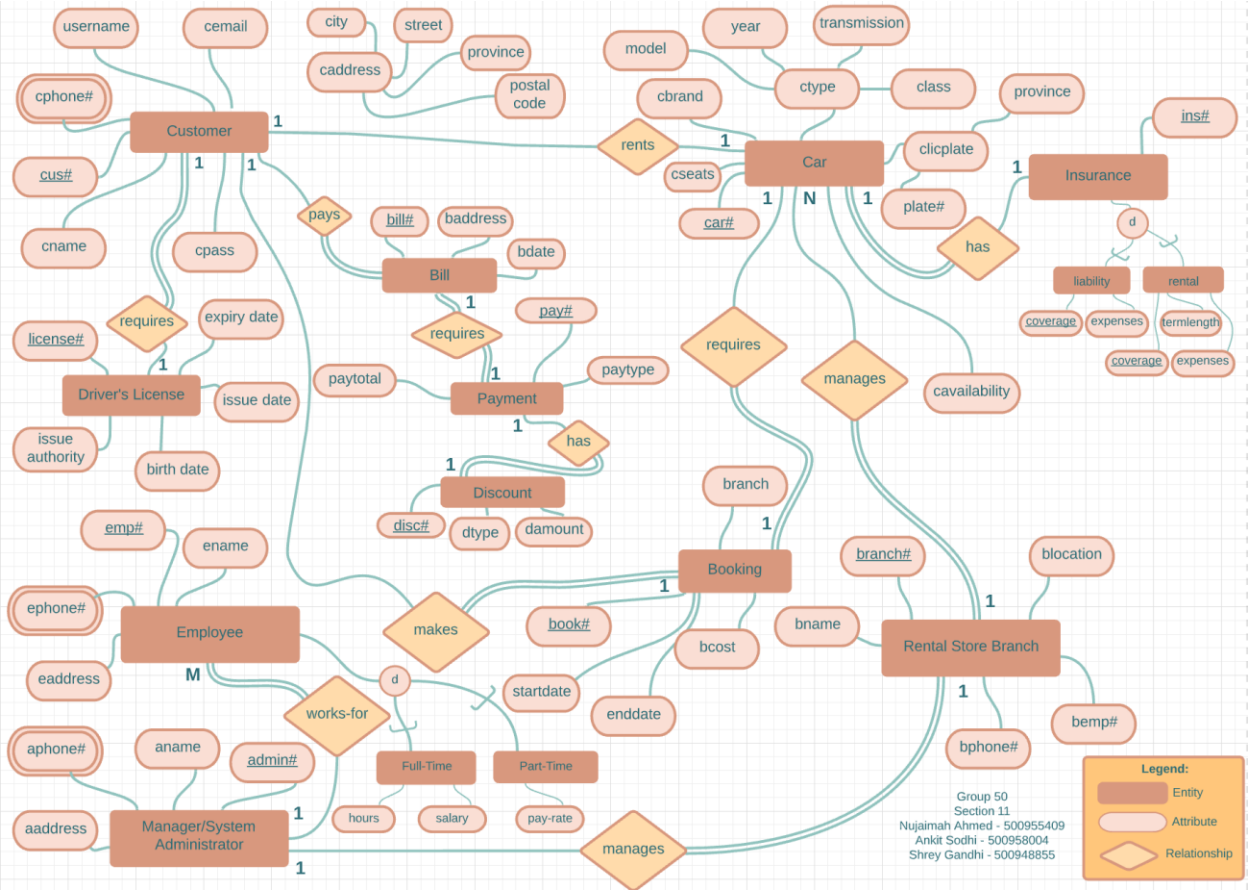
Table of Contents

1. Application Description

Our application is a Car Rental Database Management System called CRENTY. The goal of our application is to make the effort in the car rental process smoother for both the customer and the car rental agency. The system is targeted towards small car rental agencies that want to expand their rental business and need to go beyond either the paper records that they may have been using or a rudimentary application system.

Our Car Rental DBMS will allow customers to rent cars online for a period of time from anywhere in the world. Customers who want to rent a car will contact the car rental agency online regarding the vehicle that they want to rent. They must provide their information to the agency such as time of rental, type of car, identification and after confirmation the rental agency will prepare the car for rental. The customer will pay the car rental company for the car and pick the car up when it is available at the agency. The customer will return the car to the agency after the rental time is over and the car will be available for reservation again on the system.

2. ER Diagram



3. Schema Design

```
DROP TABLE Admin;
DROP TABLE RentalStoreBranch;
DROP TABLE Booking;
DROP TABLE PartTime;
DROP TABLE FullTime;
DROP TABLE Employee;
DROP TABLE DriversLicense;
DROP TABLE Discount;
DROP TABLE Bill;
DROP TABLE Payment;
DROP TABLE Rental;
DROP TABLE Liability;
DROP TABLE Car;
DROP TABLE Customer;

CREATE TABLE Customer(
   CusNumber     NUMBER PRIMARY KEY,
   CusName       VARCHAR2(50) NOT NULL,
   CusEmail      VARCHAR2(250) UNIQUE NOT NULL,
   CusUsername   VARCHAR2(20) UNIQUE NOT NULL,
   CusPassword   VARCHAR2(20) NOT NULL,
   CusPhone      VARCHAR2(12) NOT NULL,
   CusCity       VARCHAR2(50) NOT NULL,
   CusStreet     VARCHAR2(50) NOT NULL,
   CusProvince   VARCHAR2(50) NOT NULL,
   CusPostalCode VARCHAR2(50) NOT NULL
   );

INSERT INTO Customer VALUES(1, 'John', 'john123@gmail.com', 'john123', '321john', '314-
675-0091', 'Toronto', 'Apple Crescent', 'Ontario', 'M1L123');
INSERT INTO Customer VALUES(92, 'Lisa', 'lisalisa@gmail.com', 'lisalisa', 'skfmfdla', '455-
330-2043', 'Toronto', 'Dundas Street', 'Ontario', 'MA8092');
INSERT INTO Customer VALUES(302, 'Lala', 'lala@gmail.com', 'laalaa', 'aifodo', '905-321-
5435', 'Saskatoon', 'Candle Drive', 'Saskatchewan', 'YEL321');
INSERT INTO Customer VALUES(4, 'Bob', 'bobbb@gmail.com', 'bob456', '123456', '341-544-
4444', 'Winnipeg', 'Oat Street', 'Manitoba', 'M4356RS');
INSERT INTO Customer VALUES(500, 'Peace', 'peace@gmail.com', 'peaceee', 'asklklfdkl',
'416-111-0321', 'Calgary', 'Banff Drive', 'Alberta', 'M1L324');
```

```
SELECT * FROM Customer;

CREATE TABLE Car(
   CarNumber   NUMBER PRIMARY KEY,
   CarSeats    NUMBER DEFAULT 2,
   CarName     VARCHAR2(50) NOT NULL,
   CarBrand    VARCHAR2(50) NOT NULL,
   CarModel    VARCHAR2(50) NOT NULL,
   CarYear     NUMBER NOT NULL CHECK (CarYear BETWEEN 2000 AND 2021),
   CarTransmission VARCHAR2(100) NOT NULL,
   CarClass        VARCHAR2(50) NOT NULL,
   CarPlateNum     NUMBER UNIQUE NOT NULL,
   CarProvince     VARCHAR2(100) NOT NULL,
   CarCity         VARCHAR2(50) NOT NULL,
   CarStreet       VARCHAR2(50) NOT NULL,
   CarPostalCode   VARCHAR2(50) NOT NULL
   );

INSERT INTO Car VALUES(1,8,'POLO', 'VW',    'SP',    '2001', 'MANUAL', 'C', '122', 'ontario',
'TORONTO','WINK','M5A2B3');
INSERT INTO Car VALUES(2,4, 'TIG', 'RENAULT', 'S',  '2002', 'AUTO', 'C', '986', 'ontario',
'TORONTO','RED','M5B2B3');
INSERT INTO Car VALUES(3,7, 'CRETA', 'HYUN','RS',  '2005', 'AUTO', 'C', '905', 'ontario',
'TORONTO','PAT','B5A2Z4');
INSERT INTO Car VALUES(4,6, 'EECO', 'SUZUKI', 'GT',  '2007', 'AUTO', 'C', '900', 'ontario',
'TORONTO','WIMP','M5A2B3');
INSERT INTO Car VALUES(5,5, 'CITY', 'HONDA', 'PR',  '2009', 'MANUAL', 'C', '89',
'ontario', 'TORONTO','WINK','F8A2B3');
SELECT * FROM Car;

CREATE TABLE Liability(
   LiabNumber      NUMBER PRIMARY KEY,
   LiabType        VARCHAR2(50) NOT NULL CHECK (LiabType = 'Liability'),
   LiabCoverage    VARCHAR2(50) NOT NULL,
   LiabExpenses    NUMBER NOT NULL
   );

INSERT INTO Liability VALUES(10, 'Liability', 'ttrg', '2000');
INSERT INTO Liability VALUES(20, 'Liability', 'tsef', '900');
INSERT INTO Liability VALUES(30, 'Liability', 'tgd', '450');
```

```
INSERT INTO Liability VALUES(40, 'Liability', 'trfv','1200');
INSERT INTO Liability VALUES(50, 'Liability', 'jo@gmail.com', '1000');
SELECT * FROM LIABILITY;

CREATE TABLE Rental(
   RentNumber      NUMBER PRIMARY KEY,
   RentType        VARCHAR2(50) NOT NULL CHECK (RentType = 'Rental'),
   RentCoverage    VARCHAR2(50) NOT NULL,
   RentExpenses    NUMBER NOT NULL,
   TermLength      VARCHAR2(20) NOT NULL
   );

INSERT INTO Rental VALUES(1, 'Rental','t', '2000',4);
INSERT INTO Rental VALUES(2, 'Rental', 't', '900',3);
INSERT INTO Rental VALUES(3, 'Rental', 't', '450',1);
INSERT INTO Rental VALUES(4, 'Rental', 't','1200',6);
INSERT INTO Rental VALUES(5, 'Rental', 'jo@gmail.com', '1000',2);
SELECT * FROM Rental;

CREATE TABLE Payment(
   PayNumber   NUMBER PRIMARY KEY,
   PayTotal    NUMBER DEFAULT 0,
   PayType     VARCHAR2(25) NOT NULL
   );

INSERT INTO Payment VALUES(1, '90', 'CAS');
INSERT INTO Payment VALUES(2, '50', 'CASHnji');
INSERT INTO Payment VALUES(3, '20', 'CH');
INSERT INTO Payment VALUES(4, '500', 'BIT');
INSERT INTO Payment VALUES(5, '60', 'ETHEREUn');
SELECT * FROM Payment;

CREATE TABLE Bill(
   BillNumber      NUMBER PRIMARY KEY,
   BillDate        DATE UNIQUE NOT NULL,
   BillProvince    VARCHAR2(100) NOT NULL,
   BillCity        VARCHAR2(50) NOT NULL,
   BillStreet      VARCHAR2(50) NOT NULL,
   BillPostalCode  VARCHAR2(50) NOT NULL
   );
```

```
INSERT INTO Bill VALUES(1, '2021-10-03', 'ontario',    'toronto', 'wink', 'M5A2B6');
INSERT INTO Bill VALUES(2, '2021-09-04', 'ontario', 'toronto',  'red', 'X4CY76');
INSERT INTO Bill VALUES(3, '2021-01-02', 'montreal','toronto',  'dart', 'P9IY70');
INSERT INTO Bill VALUES(4, '2021-10-01', 'quebec', 'toronto',  'farm', 'M5A2B7');
INSERT INTO Bill VALUES(5, '2021-06-07', 'china', 'toronto',  'ace', 'S5F7K1');
SELECT * FROM Bill;

CREATE TABLE Discount(
   DiscNumber      NUMBER PRIMARY KEY,
   DiscAmount      NUMBER DEFAULT 0,
   DiscType        VARCHAR2(25) NOT NULL
   );

INSERT INTO Discount VALUES(1, '90', 'CASH');
INSERT INTO Discount VALUES(2, '50', 'CASH');
INSERT INTO Discount VALUES(3, '20', 'CHEQUE');
INSERT INTO Discount VALUES(4, '500', 'CASH');
SELECT * FROM Discount;

CREATE TABLE DriversLicense(
   LicenseNum      NUMBER PRIMARY KEY,
   BirthDate       VARCHAR2(30) UNIQUE NOT NULL,
   IssueDate       VARCHAR2(30) UNIQUE NOT NULL,
   ExpiryDate      VARCHAR2(30) UNIQUE NOT NULL,
   IssueAuthority  VARCHAR2(20) NOT NULL
   );

INSERT INTO DriversLicense VALUES(123, '12-11-2003', '11-10-2019', '11-10-2024',
'abdef');
INSERT INTO DriversLicense VALUES(10, '11-07-2000', '10-05-2017', '10-05-2022', 'abdef');
INSERT INTO DriversLicense VALUES(1596, '09-04-1994', '06-11-2012', '06-11-2017',
'abdef');
INSERT INTO DriversLicense VALUES(2, '08-02-1991', '03-02-2009', '03-02-2014', 'abdef');
INSERT INTO DriversLicense VALUES(4, '25-04-1985', '13-01-2000', '01-13-2005', 'abdef');
SELECT * FROM DriversLicense;

CREATE TABLE FullTime(
   EmpNumber   NUMBER PRIMARY KEY,
   EmpType     VARCHAR2(50) NOT NULL CHECK (EmpType = 'Full Time'),
```

```
   EmpPhone    VARCHAR2(12) NOT NULL ,
   EmpName     VARCHAR2(50) UNIQUE NOT NULL,
   EmpCity     VARCHAR2(50) NOT NULL,
   EmpStreet   VARCHAR2(50) NOT NULL,
   EmpProvince   VARCHAR2(50) NOT NULL,
   EmpPostalCode VARCHAR2(50),
   EmpSalary     NUMBER NOT NULL
   );

INSERT INTO FullTime VALUES(000006, 'Full Time', '416-609-3241', 'Jane', 'toronto', 'shrine
place', 'ontario', 'w1n9s5', 20000);
INSERT INTO FullTime VALUES(000007, 'Full Time', '416-020-2090', 'George', 'ajax',
'buckingham avenue', 'ontario', 'a3m4v0', 15000);
INSERT INTO FullTime VALUES(000008, 'Full Time', '647-741-9216', 'Skye', 'barrie',
'vineyard road', 'ontario', 'p5qb8d', 30000);
INSERT INTO FullTime VALUES(000009, 'Full Time', '647-332-1009', 'Henry', 'brampton',
'bloomberg crescent', 'ontario', 'j9g4q3', 45000);
INSERT INTO FullTime VALUES(000010, 'Full Time', '416-129-1001', 'Louise', 'quebec city',
'rue du petit-champlain', 'quebec', 'g6n39l', 60000);
SELECT * FROM FullTime;

CREATE TABLE PartTime(
   EmpNumber   NUMBER PRIMARY KEY,
   EmpType       VARCHAR2(50) NOT NULL CHECK (EmpType = 'Part Time'),
   EmpPhone    VARCHAR2(12) NOT NULL,
   EmpName     VARCHAR2(50) UNIQUE NOT NULL,
   EmpCity     VARCHAR2(50) NOT NULL,
   EmpStreet   VARCHAR2(50) NOT NULL,
   EmpProvince   VARCHAR2(50) NOT NULL,
   EmpPostalCode VARCHAR2(50) NOT NULL,
   EmpPayRate    NUMBER NOT NULL,
   EmpHours      NUMBER NOT NULL
   );

INSERT INTO PartTime VALUES(000001, 'Part Time', '416-827-4231', 'Sally', 'toronto', 'shrine
place', 'ontario', 'w3k4j8', 12, 15);
INSERT INTO PartTime VALUES(000002, 'Part Time', '416-200-2900', 'Angelo', 'ajax',
'buckingham avenue', 'ontario', 'b1d3q8', 15, 12);
INSERT INTO PartTime VALUES(000003, 'Part Time', '647-111-6291', 'Maya', 'barrie',
'vineyard road', 'ontario', 'x4b0n4', 20, 13);
```

```
INSERT INTO PartTime VALUES(000004, 'Part Time', '647-233-1900', 'Ben', 'brampton',
'bloomberg crescent', 'ontario', 'n4l3q8', 28, 12);
INSERT INTO PartTime VALUES(000005, 'Part Time', '416-291-0011', 'Fayette', 'quebec city',
'rue du petit-champlain', 'quebec', 'r4h8d3', 26, 15);
SELECT * FROM PartTime;

CREATE TABLE Booking(
   BookNum     NUMBER PRIMARY KEY,
   BookCost    NUMBER DEFAULT 0,
   StartDate   VARCHAR2(30) NOT NULL,
   EndDate     VARCHAR2(30) NOT NULL,
   Branch      NUMBER REFERENCES RentalStoreBranch(BranchNum) ON DELETE
CASCADE
   );

INSERT INTO Booking VALUES(1, 20, '08-02-2001', '08-04-2001', 3);
INSERT INTO Booking VALUES(2, 20, '08-02-1992', '08-03-1992', 7);
INSERT INTO Booking VALUES(3, 20, '08-03-2020', '08-05-2020', 11);
INSERT INTO Booking VALUES(4, 20, '01-06-2021', '01-10-2021', 18);
INSERT INTO Booking VALUES(5, 30, '01-06-2021', '01-10-2021', 25);
SELECT * FROM Booking;

CREATE TABLE RentalStoreBranch(
   BranchNum        NUMBER PRIMARY KEY,
   BranchName       VARCHAR2(50) NOT NULL,
   BranchPhone      VARCHAR2(12) NOT NULL,
   NumOfEmp         NUMBER DEFAULT 0,
   BranchCity       VARCHAR2(50) NOT NULL,
   BranchStreet     VARCHAR2(50) NOT NULL,
   BranchProvince   VARCHAR2(50) NOT NULL,
   BranchPostalCode VARCHAR2(50) NOT NULL
   );

INSERT INTO RentalStoreBranch VALUES(3, 'Michael', '416-493-2215', 10, 'toronto', 'shrine
place', 'ontario', 'f6l7b9');
INSERT INTO RentalStoreBranch VALUES(7, 'Aditi', '416-672-6720', 100, 'ajax', 'buckingham
avenue', 'ontario', 'd8c8s1');
INSERT INTO RentalStoreBranch VALUES(11, 'Samuel', '647-493-2215', 35, 'barrie', 'vineyard
road', 'ontario', 'm5h0g5');
```

```
INSERT INTO RentalStoreBranch VALUES(18, 'Alison', '647-395-3307', 40, 'brampton',
'bloomberg crescent', 'ontario', 'b2l5f3');
INSERT INTO RentalStoreBranch VALUES(25, 'Pierre', '819-310-9953', 60, 'quebec city', 'rue
du petit-champlain', 'ontario', 'z1e3a6');
SELECT * FROM RentalStoreBranch;

CREATE TABLE Admin(
   AdNum        NUMBER PRIMARY KEY,
   AdName       VARCHAR2(50) NOT NULL,
   AdPhone      VARCHAR2(12) NOT NULL,
   AdCity       VARCHAR2(50) NOT NULL,
   AdStreet     VARCHAR2(50) NOT NULL,
   AdProvince   VARCHAR2(50) NOT NULL,
   AdPostalCode VARCHAR2(50) NOT NULL
   );

INSERT INTO Admin VALUES(1, 'Jake', '647-455-2342', 'Ottawa', 'Pine Crescent', 'Ontario',
'MA745l');
INSERT INTO Admin VALUES(34, 'Pop', '416-342-6492', 'Brandon', 'pine crescent', 'Manitoba',
'M1L432');
INSERT INTO Admin VALUES(1540, 'Kyle', '685-092-5478', 'Regina', 'pine crescent',
'Saskatchewan', 'M58A06');
INSERT INTO Admin VALUES(2, 'Norman', '312-238-3402', 'Edmonton', 'pine crescent',
'Alberta', 'M3T654');
INSERT INTO Admin VALUES(3, 'Karen', '239-455-2342', 'Windsor', 'pine crescent', 'Ontario',
'MYA064');
SELECT * FROM Admin;
```

4. Simple Queries

```
SELECT CusName, CusProvince FROM Customer
GROUP BY CusProvince, CusName
ORDER BY CusProvince ASC;
/*Displays various selected columns grouped by  CusProvince, CusName ordered in ascending
order of CusProvince */
```

$\pi_{CusName,\ CusProvince}$ (Customer)

```
SELECT CarNumber, CarName, CarBrand, CarTransmission FROM Car
WHERE CarBrand = 'HONDA'
AND EXISTS
```

```
(SELECT * FROM Car c
WHERE c.CarTransmission = 'MANUAL')
GROUP BY CarNumber, CarName, CarBrand, CarTransmission
ORDER BY CarNumber;
/* Checks the existence of a car with a manual transmission and if found, displays only the car
brand HONDA with manual transmission. Grouped according to various columns and ordered*/
```

$\pi_{\text{CarNumber, CarName, CarBrand, CarTransmission}} (\sigma_{\text{CarBrand = 'HONDA'}} \text{ AND } \sigma_{\text{c.CarTransmission = 'MANUAL'}} (\text{Car}))$

```
SELECT COUNT(LiabType) FROM Liability
ORDER BY LiabNumber ASC;
/*Displays the number of different types of liability type in ascending order of liabNumber */
```

$F_{\text{COUNT LiabType}} (\text{Liability})$

```
SELECT RentNumber, RentType, RentExpenses, TermLength FROM Rental
    WHERE RentExpenses >=950 AND
    TermLength >=3
GROUP BY RentNumber, RentType, RentExpenses, TermLength
ORDER BY RentNumber ASC;
/*Displays selected columns where rent is greater than 950 and term length is greater than 3
grouped and then displayed in ascending order */
```

$\pi_{\text{RentNumber, RentType, RentExpenses, TermLength}} (\sigma_{\text{RentExpenses} \geq 950} \text{ AND } \sigma_{\text{TermLength} \geq 3} (\text{Rental}))$

```
SELECT 'Average Payment is', AVG(PayTotal)
FROM Payment group by 'Average Payment is';
/*Displays the average value of the payment  */
```

$F_{\text{AVERAGE PayTotal}} (\text{Payment})$

```
SELECT 'Bill #', BillNumber, ' was purchased on: ', BillDate
FROM Bill;
   /*Displays various selected columns */
```

$\pi_{\text{BillNumber, BillDate}} (\text{Bill})$

```
SELECT *
FROM Discount
WHERE DiscAmount >= 20
ORDER BY DiscAmount DESC;
/*Displays all columns where Discamount is greater than 20 and descending order of amount*/
```

$(\sigma_{\text{DiscAmount} \geq 20} (\text{Discount}))$

```
SELECT LicenseNum, IssueDate, ExpiryDate, IssueAuthority FROM DriversLicense
GROUP BY LicenseNum, IssueDate, ExpiryDate, IssueAuthority
ORDER BY LicenseNum DESC;
/*Displays various selected various columns grouped accordingly and displayed in descending
order of LicenseNumber */
```

$\pi_{\text{LicenseNum, IssueDate, ExpiryDate ,Issueauthority}}$ ((DriversLicense))

```
SELECT *
FROM FullTime
WHERE (EmpProvince = 'Ontario'
AND EmpSalary >= 20000);
   /*Displays all columns where province is 'Ontario' and have a salary greater than 20000*/
```

$(\sigma_{\texttt{EmpSalary} \geq 20000 \texttt{ AND EmpProvince='Ontario'}}$ (Fulltime))

```
SELECT 'Minimum Pay Rate is', MIN(EmpPayRate)
FROM PartTime;
/*Displays minimum payrate */
```

$F_{\text{MINIMUM EmpPayRate}}$ (PartTime)

```
SELECT DISTINCT BranchName FROM RentalStoreBranch
WHERE BranchProvince = 'ontario'
ORDER BY BranchName;
/*Displays only unique branchnames where branch province is 'Ontario' */
```

$\pi_{\text{BranchName}}$ ($\sigma_{\text{ BranchProvince='Ontario'}}$ (RentalStoreBranch))

```
SELECT Booking.BookNum, BookCost, PayTotal, PayType, Payment.PayNumber
FROM Booking, Payment
WHERE Booking.BookCost = Payment.PayTotal
GROUP BY Booking.BookNum, BookCost, PayTotal, PayType, Payment.PayNumber
ORDER BY Booking.BookNum ASC;
/*This is a joint query. Displays selected columns where booking cost and total payment are the
same. Grouped according to various columns and displayed in ascending order of booknumber */
```

Booking $\bowtie$ $\pi_{\text{BookCost, PayTotal, Booking.BookNum, PayType, Payment.PayNumber}}$ ($\sigma_{\text{Booking.BookCost = Payment.PayTotal}}$)Payment

```
SELECT *
FROM  Admin
WHERE AdName LIKE 'P%'
ORDER BY AdName DESC;
```

/*Displays the name of people with name starting with either p or n and ordered in descending order */

$\sigma_{\text{AdName LIKE 'P\%'}}$ (Admin)

```
SELECT Customer.CusNumber, Customer.CusName, DriversLicense.BirthDate,
DriversLicense.LicenseNum
FROM Customer, DriversLicense
WHERE Customer.CusNumber = DriversLicense.LicenseNum
AND Customer.CusProvince = 'Ontario'
ORDER BY CusNumber DESC;
```

Customer $\bowtie \pi$ Customer.CusNumber, Customer.CusName, DriversLicense.Birthdate, Driverslicense.LicenseNum

(σ Customer.CusNumber = DriversLicense . LicenseNum AND Customer.CusProvince = "Ontario") DriversLicense

5. Advanced Queries

```
 SELECT CarNumber, CarName, CarBrand, CarTransmission FROM Car
WHERE CarBrand = 'HONDA'
AND EXISTS
(SELECT * FROM Car c
WHERE c.CarTransmission = 'MANUAL')
GROUP BY CarNumber, CarName, CarBrand, CarTransmission
ORDER BY CarNumber;
```

$\pi_{\text{CarNumber, CarName, CarBrand, CarTransmission}}(\sigma_{\text{CarBrand = 'HONDA'}}$ AND $\sigma_{\text{c.CarTransmission = 'MANUAL'}}$(Car))

```
SELECT DiscNumber, DiscAmount, DiscType FROM Discount
WHERE DiscAmount BETWEEN 10 AND 30
GROUP BY DiscNumber, DiscAmount, DiscType
ORDER BY DiscAmount DESC;
```

$\pi_{\text{DiscNumber, DiscAmount, DiscType}}$ ($\sigma_{\text{DiscAmount} \geq 10}$ AND $\sigma_{\text{DiscAmount} < 30}$ (Discount))

```
SELECT Booking.BookNum, BookCost, PayTotal, PayType, Payment.PayNumber
FROM Booking, Payment
WHERE Booking.BookCost = Payment.PayTotal
GROUP BY Booking.BookNum, BookCost, PayTotal, PayType, Payment.PayNumber
ORDER BY Booking.BookNum ASC;
```

Booking $\bowtie \pi$ Booking.BookNum, BookCost, PayTotal, PayType ,Payment. PayNumber,

(σ Booking . BookCost = Payment . PayTotal) Payment

```
SELECT Customer.CusNumber, Customer.CusName, DriversLicense.BirthDate,
DriversLicense.LicenseNum
FROM Customer, DriversLicense
WHERE Customer.CusNumber = DriversLicense.LicenseNum
AND Customer.CusProvince = 'Ontario'
GROUP BY Customer.CusNumber, Customer.CusName, DriversLicense.BirthDate,
DriversLicense.LicenseNum
ORDER BY CusNumber DESC;
```

$$\text{Customer} \bowtie \pi_{\text{Customer.CusNumber, Customer.CusName, DriversLicense.BirthDate, DriversLicense.LicenseNum}}(\sigma_{\text{Customer.CusNumer = DriversLicense.LicenseNum AND Customer.CusProvince = 'Ontario'}}) \text{DriversLicense}$$

```
SELECT BranchNum, BranchName, BranchCity FROM RentalStoreBranch
WHERE BranchName = 'Crenty Scarborough'
AND NOT EXISTS
(SELECT * FROM RentalStoreBranch r
WHERE r.BranchCity = 'Ottawa')
GROUP BY BranchNum, BranchName, BranchCity;
```

$$\pi_{\text{BranchNum, BranchName, BranchCity}} (\sigma_{\text{BranchName='Crenty Scarborough'}} (\text{RentalStoreBranch}))$$

```
SELECT Admin.AdCity, Admin.AdName, RentalStoreBranch.BranchCity,
RentalStoreBranch.BranchName, RentalStoreBranch.NumOfEmp
FROM Admin, RentalStoreBranch
WHERE Admin.AdCity = RentalStoreBranch.BranchCity
GROUP BY Admin.AdCity, Admin.AdName, RentalStoreBranch.BranchCity,
RentalStoreBranch.BranchName, RentalStoreBranch.NumOfEmp
ORDER BY AdName DESC;
```

$$\pi_{\text{Admin.AdCity, Admin.AdName, RentalStoreBranch.BranchCity, RentalStoreBranch.BranchName, RentalStoreBranch.NumOfEmp}} (\sigma_{\text{Admin.AdCity = RentalStoreBranch.BranchCity}} (\text{Admin, RentalStoreBranch}))$$

```
SELECT EmpNumber from FullTime
UNION ALL
Select EmpNumber from PartTime where EmpProvince = 'Ontario'
GROUP BY EmpNumber;
```

$$\text{FullTime} \bowtie_{\text{EmpNumber = EmpNumber}} (\sigma_{\text{EmpProvince = 'Ontario'}} (\text{PartTime})) \text{PartTime}$$

6. Unix Shell Implementation

menu.sh

```
#!/bin/sh
MainMenu()
{

 while [ "$CHOICE" != "START" ]
 do
 clear
 echo
"================================================================="
 echo "| Oracle All Inclusive Tool
|"
 echo "| Main Menu - Select Desired Operation(s):
|"
 echo "| <CTRL-Z Anytime to Enter Interactive CMD Prompt>
|"
 echo "-----------------------------------------------------------
----"
 echo " $IS_SELECTEDM M) View Manual"
 echo " "
 echo " $IS_SELECTED1 1) Drop Tables"
 echo " $IS_SELECTED2 2) Create Tables"
 echo " $IS_SELECTED3 3) Populate Tables"
 echo " $IS_SELECTED4 4) Query Tables"
 echo " "
 echo " $IS_SELECTEDX X) Force/Stop/Kill Oracle DB"
 echo " "
 echo " $IS_SELECTEDE E) End/Exit"
 echo "Choose: "
 read CHOICE
 if [ "$CHOICE" == "0" ]
 then
 echo "Nothing Here"
 elif [ "$CHOICE" == "1" ]
 then
 bash drop_tables.sh
 Pause
 elif [ "$CHOICE" == "2" ]
 then
 bash create_tables.sh
```

```
 Pause
 elif [ "$CHOICE" == "3" ]
 then
 bash populate_tables.sh
 Pause
 elif [ "$CHOICE" == "4" ]
 then
 bash queries.sh
Pause

 elif [ "$CHOICE" == "E" ]
 then
 exit
 fi
 done
}
#--COMMENTS BLOCK--
# Main Program
#--COMMENTS BLOCK--
ProgramStart()
{
 StartMessage
 while [ 1 ]
 do
 MainMenu

 done
}
ProgramStart
```

drop_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
“n88ahmed/10295409@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ry
erson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
DROP TABLE Admin CASCADE CONSTRAINTS;
DROP TABLE RentalStoreBranch CASCADE CONSTRAINTS;
DROP TABLE Booking CASCADE CONSTRAINTS;
DROP TABLE PartTime CASCADE CONSTRAINTS;
```

```
DROP TABLE FullTime CASCADE CONSTRAINTS;
DROP TABLE DriversLicense CASCADE CONSTRAINTS;
DROP TABLE Discount CASCADE CONSTRAINTS;
DROP TABLE Bill CASCADE CONSTRAINTS;
DROP TABLE Payment CASCADE CONSTRAINTS;
DROP TABLE Rental CASCADE CONSTRAINTS;
DROP TABLE Liability CASCADE CONSTRAINTS;
DROP TABLE Car CASCADE CONSTRAINTS;
DROP TABLE Customer CASCADE CONSTRAINTS;
exit;
EOF
```

create_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
“username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.rye
rson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

CREATE TABLE Customer(
        CusNumber     NUMBER PRIMARY KEY,
        CusName       VARCHAR2(50) NOT NULL,
        CusEmail      VARCHAR2(250) UNIQUE NOT NULL,
        CusUsername   VARCHAR2(20) UNIQUE NOT NULL,
        CusPassword   VARCHAR2(20) NOT NULL,
        CusPhone      VARCHAR2(12) NOT NULL,
        CusCity       VARCHAR2(50) NOT NULL,
        CusStreet     VARCHAR2(50) NOT NULL,
        CusProvince   VARCHAR2(50) NOT NULL,
        CusPostalCode VARCHAR2(50) NOT NULL
        );

CREATE TABLE Car(
        CarNumber   NUMBER PRIMARY KEY,
        CarSeats    NUMBER DEFAULT 2,
        CarName     VARCHAR2(50) NOT NULL,
        CarBrand    VARCHAR2(50) NOT NULL,
        CarModel    VARCHAR2(50) NOT NULL,
        CarYear     NUMBER NOT NULL CHECK (CarYear BETWEEN 2000 AND 2021),
```

```
    CarTransmission VARCHAR2(100) NOT NULL,
    CarClass        VARCHAR2(50) NOT NULL,
    CarPlateNum     NUMBER UNIQUE NOT NULL,
    CarProvince     VARCHAR2(100) NOT NULL,
    CarCity         VARCHAR2(50) NOT NULL,
    CarStreet       VARCHAR2(50) NOT NULL,
    CarPostalCode   VARCHAR2(50) NOT NULL
    );

CREATE TABLE Liability(
    LiabNumber      NUMBER PRIMARY KEY,
    LiabType        VARCHAR2(50) NOT NULL CHECK (LiabType = 'Liability'),
    LiabCoverage    VARCHAR2(50) NOT NULL,
    LiabExpenses    NUMBER NOT NULL
    );

CREATE TABLE Rental(
    RentNumber      NUMBER PRIMARY KEY,
    RentType        VARCHAR2(50) NOT NULL CHECK (RentType = 'Rental'),
    RentCoverage    VARCHAR2(50) NOT NULL,
    RentExpenses    NUMBER NOT NULL,
    TermLength      VARCHAR2(20) NOT NULL
    );

CREATE TABLE Payment(
    PayNumber       NUMBER PRIMARY KEY,
    PayTotal        NUMBER DEFAULT 0,
    PayType         VARCHAR2(25) NOT NULL
    );

CREATE TABLE Bill(
    BillNumber      NUMBER PRIMARY KEY,
    BillDate        VARCHAR(20) UNIQUE NOT NULL,
    BillProvince    VARCHAR2(100) NOT NULL,
    BillCity        VARCHAR2(50) NOT NULL,
    BillStreet      VARCHAR2(50) NOT NULL,
    BillPostalCode  VARCHAR2(50) NOT NULL
    );

CREATE TABLE Discount(
```

```
    DiscNumber   NUMBER PRIMARY KEY,
    DiscAmount   NUMBER DEFAULT 0,
    DiscType     VARCHAR2(25) NOT NULL
    );

CREATE TABLE DriversLicense(
    LicenseNum   NUMBER PRIMARY KEY,
    BirthDate    VARCHAR2(30) UNIQUE NOT NULL,
    IssueDate    VARCHAR2(30) UNIQUE NOT NULL,
    ExpiryDate   VARCHAR2(30) UNIQUE NOT NULL,
    IssueAuthority  VARCHAR2(20) NOT NULL
    );

CREATE TABLE FullTime(
    EmpNumber   NUMBER PRIMARY KEY,
    EmpType      VARCHAR2(50) NOT NULL CHECK (EmpType = 'Full Time'),
    EmpPhone     VARCHAR2(12) NOT NULL ,
    EmpName      VARCHAR2(50) UNIQUE NOT NULL,
    EmpCity      VARCHAR2(50) NOT NULL,
    EmpStreet    VARCHAR2(50) NOT NULL,
    EmpProvince   VARCHAR2(50) NOT NULL,
    EmpPostalCode VARCHAR2(50),
    EmpSalary    NUMBER NOT NULL
    );

CREATE TABLE PartTime(
    EmpNumber   NUMBER PRIMARY KEY,
    EmpType      VARCHAR2(50) NOT NULL CHECK (EmpType = 'Part Time'),
    EmpPhone     VARCHAR2(12) NOT NULL,
    EmpName      VARCHAR2(50) UNIQUE NOT NULL,
    EmpCity      VARCHAR2(50) NOT NULL,
    EmpStreet    VARCHAR2(50) NOT NULL,
    EmpProvince   VARCHAR2(50) NOT NULL,
    EmpPostalCode VARCHAR2(50) NOT NULL,
    EmpPayRate   NUMBER NOT NULL,
    EmpHours     NUMBER NOT NULL
    );

CREATE TABLE RentalStoreBranch(
    BranchNum    NUMBER PRIMARY KEY,
```

```
        BranchName          VARCHAR2(50) NOT NULL,
        BranchPhone  VARCHAR2(12) NOT NULL,
        NumOfEmp            NUMBER DEFAULT 0,
        BranchCity    VARCHAR2(50) NOT NULL,
        BranchStreet  VARCHAR2(50) NOT NULL,
        BranchProvince   VARCHAR2(50) NOT NULL,
        BranchPostalCode VARCHAR2(50) NOT NULL
        );

CREATE TABLE Booking(
        BookNum      NUMBER PRIMARY KEY,
        BookCost     NUMBER DEFAULT 0,
        StartDate   VARCHAR2(30) NOT NULL,
        EndDate      VARCHAR2(30) NOT NULL,
        Branch       NUMBER REFERENCES RentalStoreBranch(BranchNum) ON DELETE
CASCADE
        );

CREATE TABLE Admin(
        AdNum        NUMBER PRIMARY KEY,
        AdName       VARCHAR2(50) NOT NULL,
        AdPhone      VARCHAR2(12) NOT NULL,
        AdCity       VARCHAR2(50) NOT NULL,
        AdStreet     VARCHAR2(50) NOT NULL,
        AdProvince   VARCHAR2(50) NOT NULL,
        AdPostalCode VARCHAR2(50) NOT NULL
        );

exit;
EOF
```

populate_tables.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
“username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.rye
rson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
```

```
INSERT INTO Customer VALUES(123, 'John', 'john123@gmail.com', 'john123', '321john',
'314-675-0091', 'Toronto', 'Apple Crescent', 'Ontario', 'M1L123');
INSERT INTO Customer VALUES(10, 'Lisa', 'lisalisa@gmail.com', 'lisalisa', 'skfmfdla', '455-
330-2043', 'Toronto', 'Dundas Street', 'Ontario', 'MA8092');
INSERT INTO Customer VALUES(1596, 'Lala', 'lala@gmail.com', 'laalaa', 'aifodo', '905-321-
5435', 'Saskatoon', 'Candle Drive', 'Saskatchewan', 'YEL321');
INSERT INTO Customer VALUES(2, 'Bob', 'bobbb@gmail.com', 'bob456', '123456', '341-544-
4444', 'Winnipeg', 'Oat Street', 'Manitoba', 'M4356RS');
INSERT INTO Customer VALUES(4, 'Peace', 'peace@gmail.com', 'peaceee', 'asklklfdkl', '416-
111-0321', 'Calgary', 'Banff Drive', 'Alberta', 'M1L324');

INSERT INTO Car VALUES(1,8,'POLO', 'VW',      'SP',     '2001', 'MANUAL', 'C', '122',
'ontario', 'TORONTO','WINK','M5A2B3');
INSERT INTO Car VALUES(2,4, 'TIG', 'RENAULT', 'S',  '2002', 'AUTO', 'C', '986', 'ontario',
'TORONTO','RED','M5B2B3');
INSERT INTO Car VALUES(3,7, 'CRETA', 'HONDA','RS',  '2005', 'MANUAL', 'C', '905',
'ontario', 'TORONTO','PAT','B5A2Z4');
INSERT INTO Car VALUES(4,6, 'EECO', 'SUZUKI', 'GT',  '2007', 'AUTO', 'C', '900', 'ontario',
'TORONTO','WIMP','M5A2B3');
INSERT INTO Car VALUES(5,5, 'CITY', 'HONDA', 'PR',  '2009', 'MANUAL', 'C', '89',
'ontario', 'TORONTO','WINK','F8A2B3');

INSERT INTO Liability VALUES(10, 'Liability', 'ttrg', '2000');
INSERT INTO Liability VALUES(20, 'Liability', 'tsef', '900');
INSERT INTO Liability VALUES(30, 'Liability', 'tgd', '450');
INSERT INTO Liability VALUES(40, 'Liability', 'trfv','1200');
INSERT INTO Liability VALUES(50, 'Liability', 'jo@gmail.com', '1000');

INSERT INTO Rental VALUES(1, 'Rental','t', '1200',4);
INSERT INTO Rental VALUES(2, 'Rental', 't', '560',3);
INSERT INTO Rental VALUES(3, 'Rental', 't', '1000',1);
INSERT INTO Rental VALUES(4, 'Rental', 't','210',6);
INSERT INTO Rental VALUES(5, 'Rental', 'jo@gmail.com', '980',2);

INSERT INTO Payment VALUES(1, '20', 'CASH');
INSERT INTO Payment VALUES(2, '200', 'CHEQUE');
INSERT INTO Payment VALUES(3, '150', 'DEBIT');
INSERT INTO Payment VALUES(4, '230', 'BIT');
INSERT INTO Payment VALUES(5, '30', 'CREDIT');
```

```
INSERT INTO Bill VALUES(1, '2021-10-03', 'Ontario', 'Toronto', 'Wink Lane', 'M5A2B6');
INSERT INTO Bill VALUES(209, '2021-09-04', 'Ontario', 'Toronto', 'Red Crescent', 'X4CY76');
INSERT INTO Bill VALUES(3111, '2021-01-02', 'Montreal', 'Toronto', 'Dart Drive', 'P9IY70');
INSERT INTO Bill VALUES(43, '2021-10-01', 'Quebec', 'Toronto', 'Farm Avenue', 'M5A2B7');
INSERT INTO Bill VALUES(523, '2021-06-07', 'China', 'Toronto', 'Ace Street', 'S5F7K1');

INSERT INTO Discount VALUES(1, '15', 'CASH');
INSERT INTO Discount VALUES(20, '50', 'CASH');
INSERT INTO Discount VALUES(34, '20', 'CHEQUE');
INSERT INTO Discount VALUES(334, '10', 'CASH');

INSERT INTO DriversLicense VALUES(123, '12-11-2003', '11-10-2019', '11-10-2024',
'MTO');
INSERT INTO DriversLicense VALUES(10, '11-07-2000', '10-05-2017', '10-05-2022', 'DMV');
INSERT INTO DriversLicense VALUES(1596, '09-04-1994', '06-11-2012', '06-11-2017',
'MTO');
INSERT INTO DriversLicense VALUES(2, '08-02-1991', '03-02-2009', '03-02-2014', 'DMV');
INSERT INTO DriversLicense VALUES(4, '25-04-1985', '13-01-2000', '01-13-2005', 'DMV');

INSERT INTO FullTime VALUES(000006, 'Full Time', '416-609-3241', 'Jane', 'Toronto',
'Shrine Place', 'Ontario', 'M5P482', 20000);
INSERT INTO FullTime VALUES(000007, 'Full Time', '416-020-2090', 'George', 'Ajax',
'Buckingham Avenue', 'Ontario', 'A4RT72', 15000);
INSERT INTO FullTime VALUES(000008, 'Full Time', '647-741-9216', 'Skye', 'Barrie',
'Vineyard Road', 'Winnipeg', 'P89T76', 30000);
INSERT INTO FullTime VALUES(000009, 'Full Time', '647-332-1009', 'Henry', 'Brampton',
'Bloomberg Crescent', 'Ontario', 'TY7R87', 45000);
INSERT INTO FullTime VALUES(000010, 'Full Time', '416-129-1001', 'Louise', 'Quebec City',
'Rue du Petit-Champlain', 'Quebec', 'AS8TY7', 60000);

INSERT INTO PartTime VALUES(000001, 'Part Time', '416-827-4231', 'Sally', 'Toronto',
'Shrine Place', 'Ontario', 'M5P482', 12, 15);
INSERT INTO PartTime VALUES(000002, 'Part Time', '416-200-2900', 'Angelo', 'Ajax',
'Buckingham Avenue', 'Ontario', 'A4RT72', 15, 12);
INSERT INTO PartTime VALUES(000003, 'Part Time', '647-111-6291', 'Maya', 'Barrie',
'Bloomberg Road', 'Ontario', 'P89T76', 20, 13);
INSERT INTO PartTime VALUES(000004, 'Part Time', '647-233-1900', 'Ben', 'Brampton',
'Bloomberg Crescent', 'Ontario', 'TY7R87', 28, 12);
INSERT INTO PartTime VALUES(000005, 'Part Time', '416-291-0011', 'Fayette', 'Quebec City',
'Rue du Petit-Champlain', 'Quebec', 'AS8TY7', 26, 15);
```

```
INSERT INTO RentalStoreBranch VALUES(3, 'Crenty Scarborough', '416-493-2215', 10,
'Ottawa', 'shrine place', 'ontario', 'f6l7b9');
INSERT INTO RentalStoreBranch VALUES(7, 'Crenty Scarborough', '416-672-6720', 100,
'Brandon', 'buckingham avenue', 'ontario', 'd8c8s1');
INSERT INTO RentalStoreBranch VALUES(11, 'missipisi', '647-493-2215', 35, 'barrie',
'Regina', 'ontario', 'm5h0g5');
INSERT INTO RentalStoreBranch VALUES(18, 'hitman', '647-395-3307', 40, 'brampton',
'bloomberg crescent', 'ontario', 'b2l5f3');
INSERT INTO RentalStoreBranch VALUES(25, 'toronto', '819-310-9953', 60, 'quebec city', 'rue
du petit-champlain', 'ontario', 'z1e3a6');

INSERT INTO Booking VALUES(1, 20, '08-02-2001', '08-04-2001', 3);
INSERT INTO Booking VALUES(2, 200, '08-02-1992', '08-03-1992', 7);
INSERT INTO Booking VALUES(3, 150, '08-03-2020', '08-05-2020', 11);
INSERT INTO Booking VALUES(4, 230, '01-06-2021', '01-10-2021', 18);
INSERT INTO Booking VALUES(5, 30, '01-06-2021', '01-10-2021', 25);

INSERT INTO Admin VALUES(1, 'Jake', '647-455-2342', 'Ottawa', 'Pine Crescent', 'Ontario',
'MA745l');
INSERT INTO Admin VALUES(34, 'Pop', '416-342-6492', 'Brandon', 'pine crescent', 'Manitoba',
'M1L432');
INSERT INTO Admin VALUES(1540, 'Pape', '685-092-5478', 'Regina', 'pine crescent',
'Saskatchewan', 'M58A06');
INSERT INTO Admin VALUES(2, 'Norman', '312-238-3402', 'Edmonton', 'pine crescent',
'Alberta', 'M3T654');
INSERT INTO Admin VALUES(3, 'Karen', '239-455-2342', 'Windsor', 'pine crescent', 'Ontario',
'MYA064');

exit;
EOF
```

queries.sh

```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64
“username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.rye
rson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
```

```
SELECT CusName, CusProvince FROM Customer
GROUP BY CusProvince, CusName
ORDER BY CusProvince ASC;

SELECT CarNumber, CarName, CarBrand, CarTransmission FROM Car
WHERE CarBrand = 'HONDA'
AND EXISTS
(SELECT * FROM Car c
WHERE c.CarTransmission = 'MANUAL')
GROUP BY CarNumber, CarName, CarBrand, CarTransmission
ORDER BY CarNumber;

SELECT COUNT(LiabType) FROM Liability
ORDER BY LiabNumber ASC;

SELECT RentNumber, RentType, RentExpenses, TermLength FROM Rental
        WHERE RentExpenses >=950 AND
        TermLength >=3
GROUP BY RentNumber, RentType, RentExpenses, TermLength
ORDER BY RentNumber ASC;

SELECT 'Average Payment is', AVG(PayTotal)
FROM Payment group by 'Average Payment is';

SELECT 'Bill #', BillNumber, ' was purchased on: ', BillDate
FROM Bill;

SELECT DiscNumber, DiscAmount, DiscType FROM Discount
WHERE DiscAmount BETWEEN 10 AND 30
GROUP BY DiscNumber, DiscAmount, DiscType
ORDER BY DiscAmount DESC;

SELECT LicenseNum, IssueDate, ExpiryDate, IssueAuthority FROM DriversLicense
GROUP BY LicenseNum, IssueDate, ExpiryDate, IssueAuthority
ORDER BY LicenseNum DESC;

SELECT *
FROM FullTime
WHERE (EmpProvince = 'Ontario'
        AND EmpSalary >= 20000);
```

```
SELECT 'Minimum Pay Rate is', MIN(EmpPayRate)
FROM PartTime;

SELECT 'Maxmimum Number of Employees is', Max(NumOfEmp)
FROM RentalStoreBranch;

SELECT Booking.BookNum, BookCost, PayTotal, PayType, Payment.PayNumber
FROM Booking, Payment
WHERE Booking.BookCost = Payment.PayTotal
GROUP BY Booking.BookNum, BookCost, PayTotal, PayType, Payment.PayNumber
ORDER BY Booking.BookNum ASC;

SELECT AdNum, AdName FROM  Admin
WHERE AdName LIKE 'P%'
OR AdName LIKE 'N%'
GROUP BY AdNum, AdName
ORDER BY AdName DESC;

SELECT Customer.CusNumber, Customer.CusName, DriversLicense.BirthDate,
DriversLicense.LicenseNum
FROM Customer, DriversLicense
WHERE Customer.CusNumber = DriversLicense.LicenseNum
AND Customer.CusProvince = 'Ontario'
GROUP BY Customer.CusNumber, Customer.CusName, DriversLicense.BirthDate,
DriversLicense.LicenseNum
ORDER BY CusNumber DESC;

SELECT DISTINCT BranchName FROM RentalStoreBranch
WHERE BranchProvince = 'ontario'
GROUP BY BranchName;

SELECT BranchNum, BranchName, BranchCity FROM RentalStoreBranch
WHERE BranchName = 'Crenty Scarborough'
AND NOT EXISTS
(SELECT * FROM RentalStoreBranch r
WHERE r.BranchCity = 'Ottawa')
GROUP BY BranchNum, BranchName,BranchCity;
```

```
SELECT Admin.AdCity, Admin.AdName, RentalStoreBranch.BranchCity,
RentalStoreBranch.BranchName, RentalStoreBranch.NumOfEmp
FROM Admin, RentalStoreBranch
WHERE Admin.AdCity = RentalStoreBranch.BranchCity
GROUP BY Admin.AdCity, Admin.AdName, RentalStoreBranch.BranchCity,
RentalStoreBranch.BranchName, RentalStoreBranch.NumOfEmp
ORDER BY AdName DESC;

SELECT EmpNumber from FullTime
UNION ALL
Select EmpNumber from PartTime where EmpProvince = 'Ontario'
GROUP BY EmpNumber;

exit;
EOF
```

7. **Functional Dependencies**

Customer(<u>CusNumber</u>, CusName, CusEmail, CusUsername, CusPassword, CusPhone, CusCity, CusStreet, CusProvince, CusPostalCode)

FD: {CusNumber → CusName
CusNumber → CusEmail
CusNumber → CusUsername
CusNumber → CusPassword
CusNumber → CusPhone
CusNumber → CusCity
CusNumber → CusStreet
CusNumber → CusProvince
CusNumber → CusPostalCode
CusEmail → CusName
CusCity → CusProvince}

Car(<u>CarNumber</u>, CarName, CarSeats, CarBrand, CarModel, CarYear, CarTransmission, CarClass, CarPlateNum, CarProvince, CarCity, CarStreet, CarPostalCode)

FD: {CarNumber → CarName
CarNumber → CarSeats

CarNumber → CarName
CarNumber → CarBrand
CarNumber → CarModel
CarNumber → CarYear
CarNumber → CarTransmission
CarNumber → CarClass
CarNumber → CarPlateNum
CarNumber → CarProvince
CarNumber → CarCity
CarNumber → CarStreet
CarNumber → CarPostalCode
CarModel → CarYear}

Liability(<u>LiabNumber</u>, LiabType, LiabCoverage, LiabExpenses)

FD: {LiabNumber → LiabType
LiabNumber → LiabCoverage
LiabNumber → LiabExpenses}

Rental(<u>RentNumber</u>, RentType, RentCoverage, RentExpenses, TermLength)

FD: {RentNumber → RentType
RentNumber → RentCoverage
RentNumber → RentExpenses
RentNumber → TermLength
RentType → RentCoverage}

Payment(<u>PayNumber</u>, PayTotal, PayType)

FD: {PayNumber → PayTotal
PayNumber → PayType}

Bill(<u>BillNumber</u>, BillDate, BillProvince, BillCity, BillStreet, BillPostalCode)

FD: {BillNumber → BillDate
BillNumber → BillProvince

BillNumber → BillCity
BillNumber → BillStreet
BillNumber → BillPostalCode
BillPostalCode → BillStreet
BillPostalCode → BillCity
BillPostalCode → BillProvince}

Discount(DiscNumber, DiscAmount, DiscType)

FD: {DiscNumber → DiscAmount
DiscNumber → DiscType}

DriversLicense(LicenseNum, BirthDate, IssueDate, ExpiryDate, IssueAuthority)

FD: {LicenseNum, DriverName → BirthDate
LicenseNum → IssueDate
LicenseNum → ExpiryDate
LicenseNum → IssueAuthority
IssueAuthority → ExpiryDate
IssueAuthority → IssueDate}

FullTime(EmpNumber, EmpName, EmpType, EmpPhone, EmpCity, EmpStreet, EmpProvince, EmpPostalCode, EmpSalary)

FD: {EmpNumber → EmpName
EmpNumber → EmpType
EmpNumber → EmpPhone
EmpNumber → EmpCity
EmpNumber → EmpStreet
EmpNumber → EmpProvince
EmpNumber → EmpPostalCode
EmpNumber → EmpSalary
EmpProvince → EmpCity}

PartTime(EmpNumber, EmpType, EmpPhone, EmpName, EmpCity, EmpStreet, EmpProvince, EmpPostalCode, EmpPayRate, EmpHours)

FD: {EmpNumber → EmpName
EmpNumber → EmpType
EmpNumber → EmpPhone
EmpNumber → EmpCity
EmpNumber → EmpStreet
EmpNumber → EmpProvince
EmpNumber → EmpPostalCode
EmpNumber → EmpPayRate
EmpNumber → EmpHours
EmpHours → EmpPayRate}

RentalStoreBranch(<u>BranchNum</u>, BranchName, BranchPhone, NumOfEmp, BranchCity, BranchStreet, BranchProvince, BranchPostalCode)

FD: {BranchNum, BranchName → BranchPhone
BranchNum → NumOfEmp
BranchNum → BranchCity
BranchNum → BranchStreet
BranchNum → BranchProvince
BranchNum → BranchPostalCode
BranchPostalCode → BranchProvince}

Booking(<u>BookNum,</u> BookCost, StartDate, EndDate, Branch)

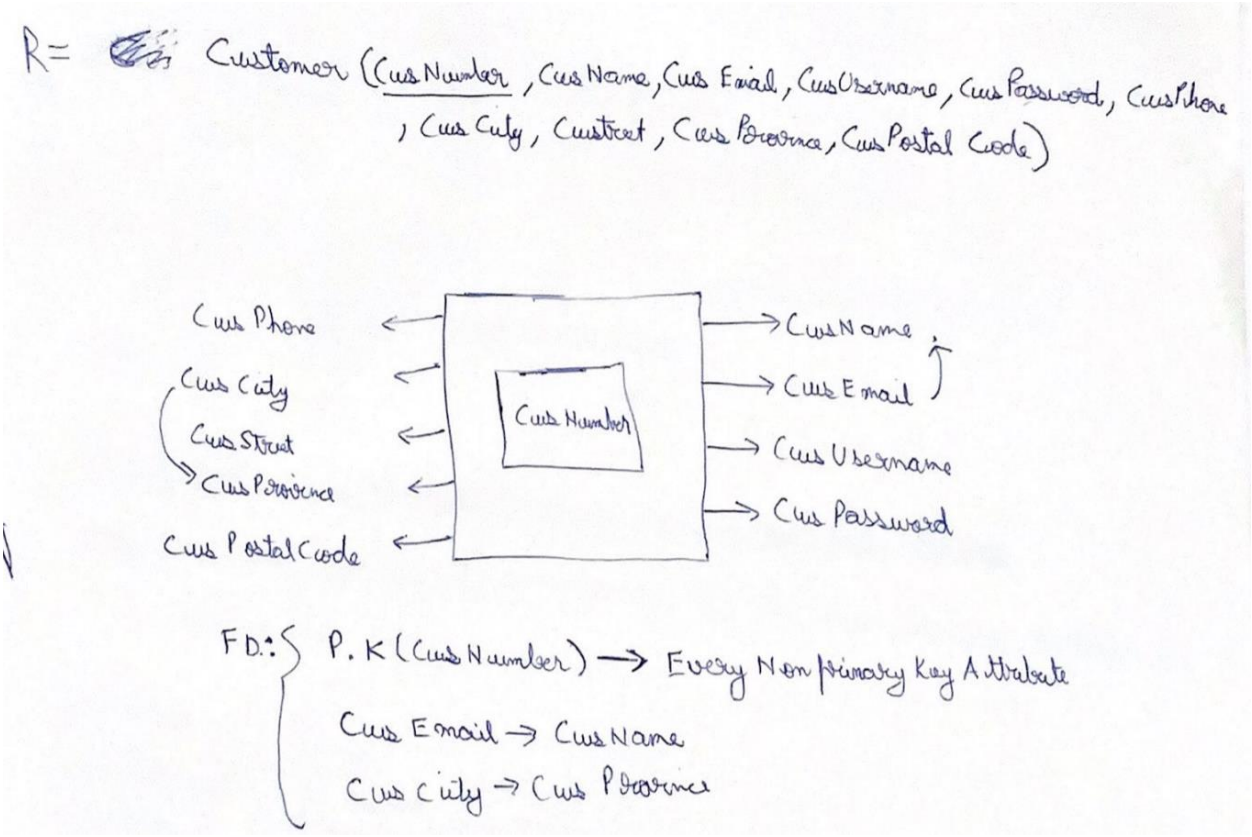
FD: {BookNum → BookCost
BookNum → StartDate
BookNum → EndDate
BookNum → Branch
BookCost → StartDate} (one-to-many relationship between booking and rental store branch, therefore there is a functional dependency)

Admin(AdNum, AdName, AdPhone, AdCity, AdStreet, AdProvince, AdPostalCode)

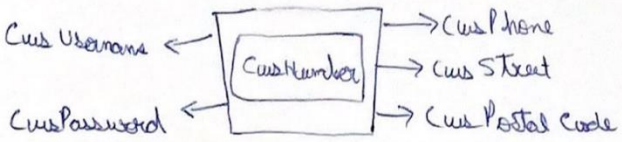
FD = {AdNum → AdName
AdNum → AdPhone

AdNum → AdCity
AdNum → AdStreet
AdNum → AdProvince
AdNum → AdPostalCode
AdName → AdPhone}

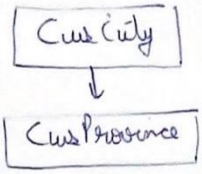
8. **Normalization: 3NF**



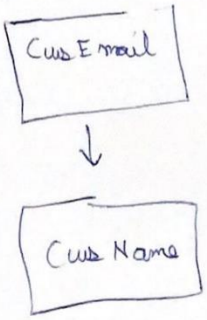
R1.1 (Cus Number , Cus Phone, Cus Street , Cus Postal Code , Cus Username, Cus Password)



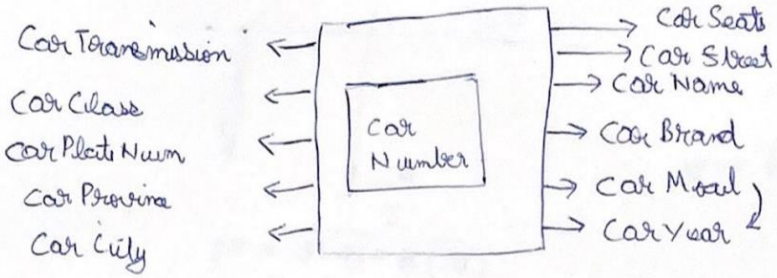
R.1.2) (Cus City, Cus Province)



R 1.3) (Cus Email, Cus Name)



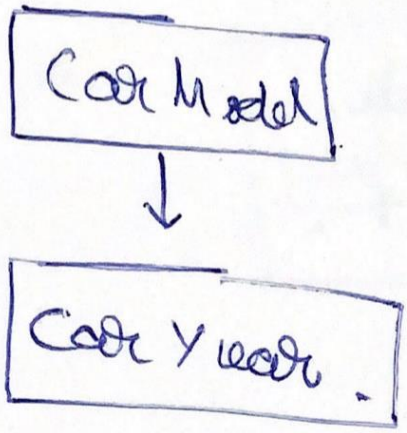
R = Car(Car Number , Car Seats , Car Name , Car Brand , Car Model , Car Year , Car Transmission , Car Class , Car Plate Num , Car Province Car City , Car Street)



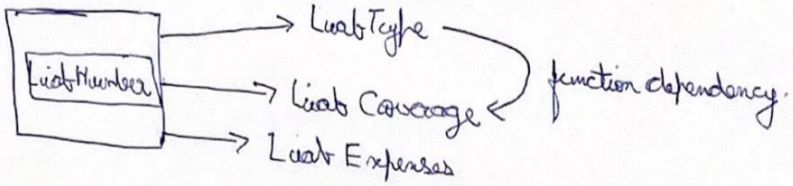
F.D : P.K (Car Number) → Every Non Primary Key Attribute

Car Model → Car Year

R1.2) (Car Model, Car Year)

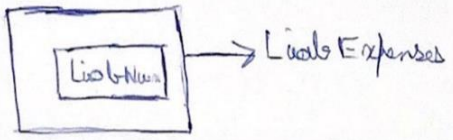


R= Liability (LiabNumber, LiabType, LiabCoverage, LiabExpenses)



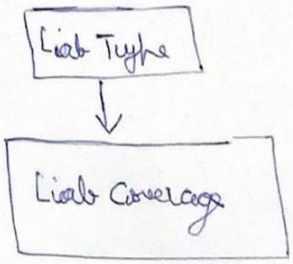
FD:
1.) LiabNumber → LiabType
2.) LiabNumber → Liab Coverage
3.) LiabNumber → Liab Expenses
4.) LiabType → Liab Coverage.

R1.1 (Liab Number, Liab Expenses)



FD: Liab Num → Liab Expenses

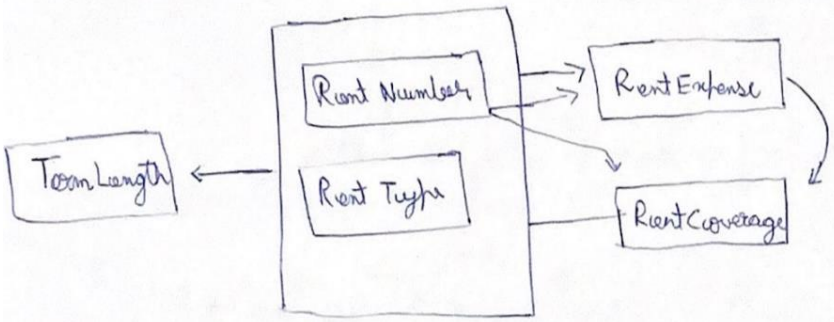
R1.2 (Liab Type, Liab Coverage)



FD: Liab Type → Liab Coverage

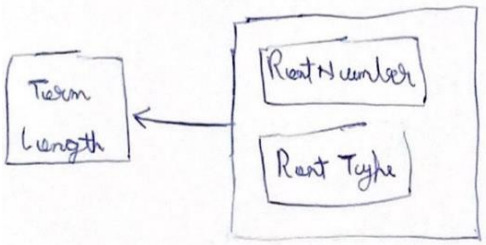
R = Rental (Rent Number, Rent Type, Rent Coverage, Rent Expenses, Term Length)

We have two primary keys (Assumption)

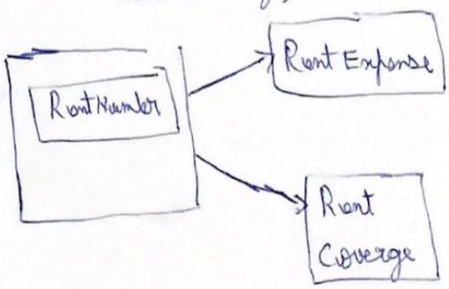


FD:
1.) Rent Number, Rent Type → Term Length, Rent Expense, Rent Coverage
2.) Rent Number → Rent Expense
3) Rent Number → Rent Coverage
4) Rent Expense → Rent Coverage.

R1.1) (Rent Number, Rent Type, Term Length)

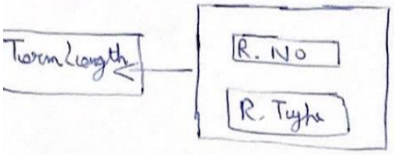


R1.2) (Rent Number, Rent Expense, Rent Coverage).



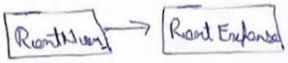
R1.1)

(Rent Number, Rent Type, Term Length)



FD { Rent No, Rent Type → Term Length

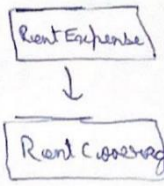
R1.2) (Rent Number, Rent Expense)



FD { Rent Number → Rent Expense

R1.3)

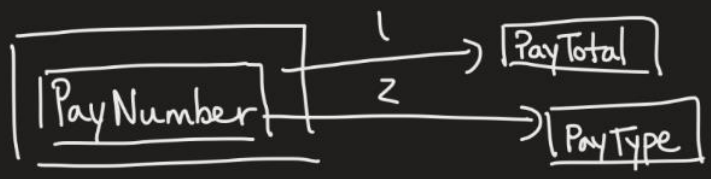
(Rent Expense, Rent Coverage).



FD { Rent Expense → Rent Coverage.

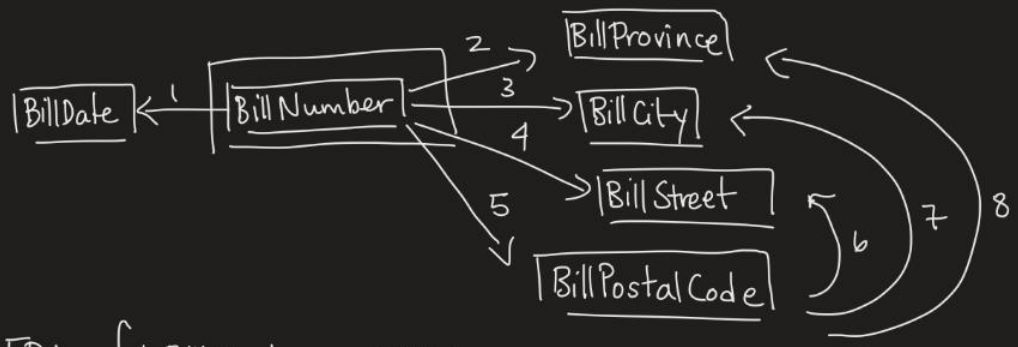
R = Payment (PayNumber, PayTotal, PayType)

1NF
2NF
3NF



FD:
1. PayNumber → PayTotal
2. PayNumber → PayType

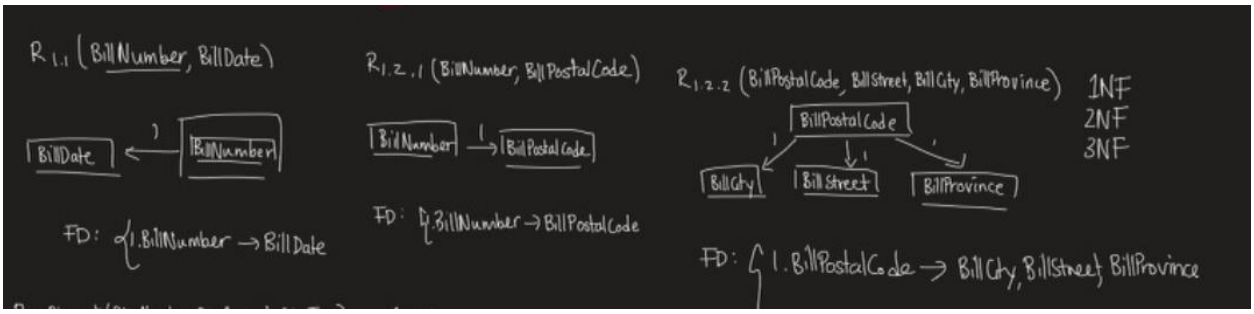
R = Bill (BillNumber, BillDate, BillProvince, BillCity, BillStreet, BillPostalCode)

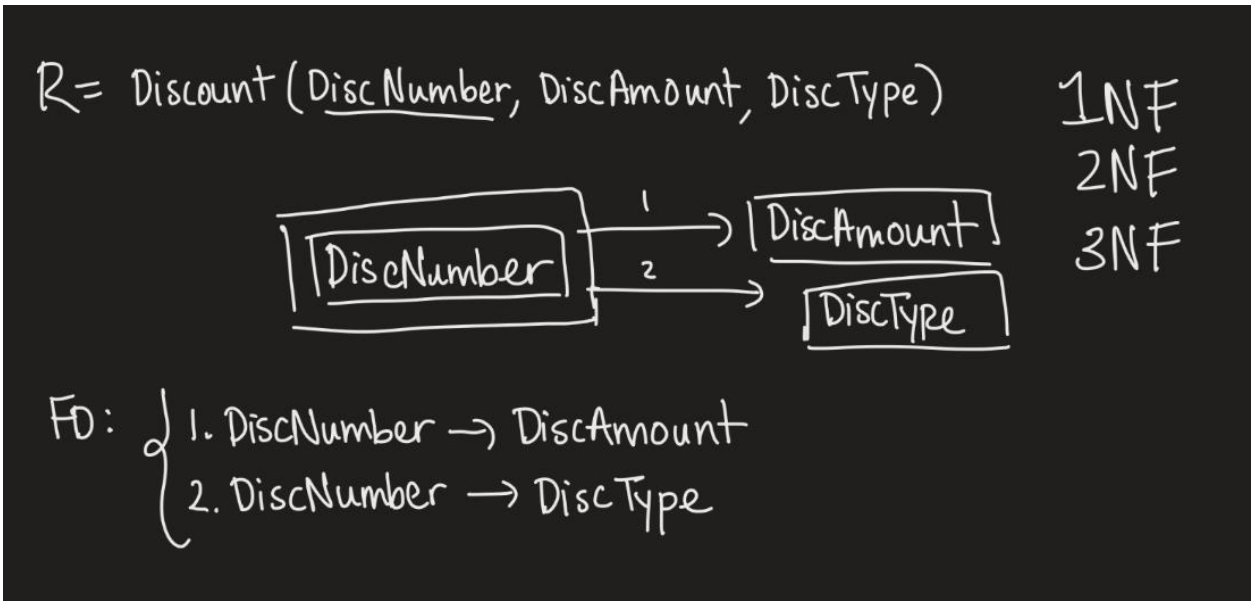


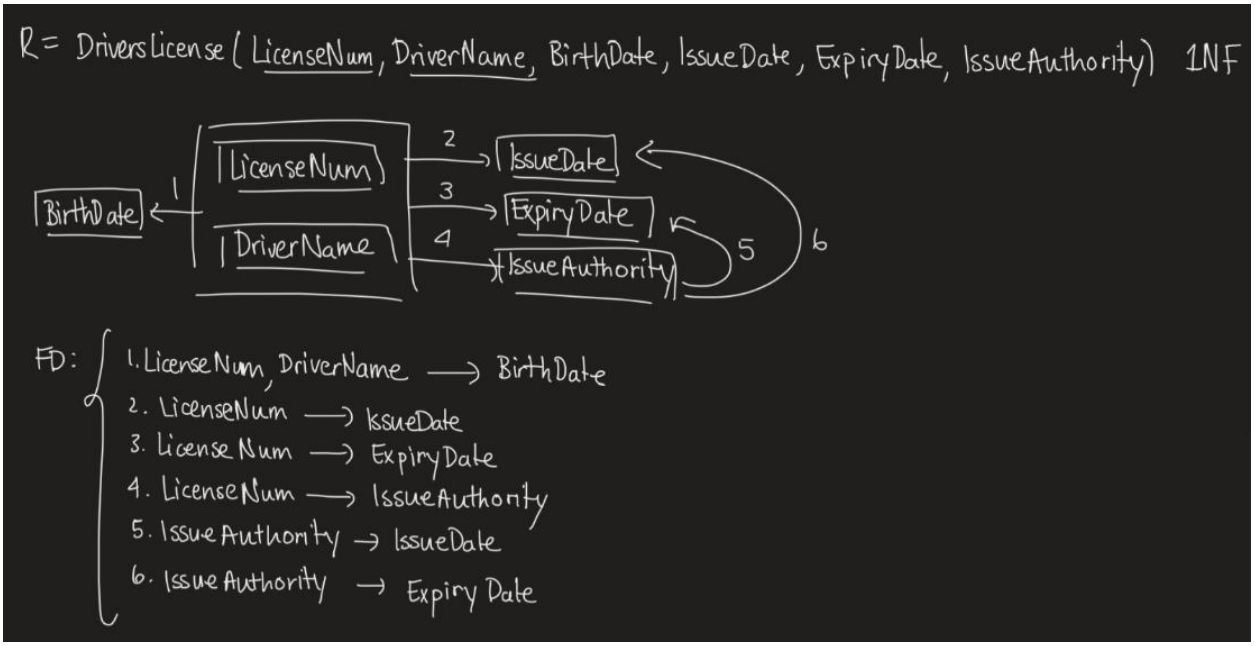
1NF
2NF

FD:
1. BillNumber → BillDate
2. BillNumber → BillProvince
3. BillNumber → BillCity
4. BillNumber → BillStreet
5. BillNumber → BillPostalCode
6. BillPostalCode → BillProvince
7. BillPostalCode → BillStreet
8. BillPostalCode → BillCity

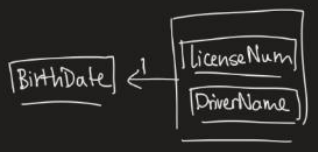
transitivity


R1.1 (BillNumber, BillDate)
BillDate
BillNumber
FD: 1. BillNumber → BillDate
R1.2.1 (BillNumber, BillPostalCode)
BillNumber
BillPostalCode
FD: 1. BillNumber → BillPostalCode
R1.2.2 (BillPostalCode, BillStreet, BillCity, BillProvince)
BillPostalCode
BillCity
BillStreet
BillProvince
FD: 1. BillPostalCode → BillCity, BillStreet, BillProvince
1NF
2NF
3NF


R= Discount (DiscNumber, DiscAmount, DiscType)
1NF
2NF
3NF
DiscNumber
DiscAmount
DiscType
1
2
FD:
1. DiscNumber → DiscAmount
2. DiscNumber → DiscType

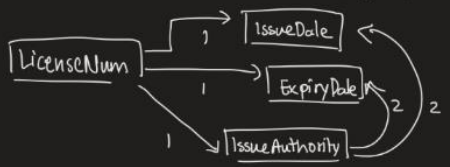

R= DriversLicense (LicenseNum, DriverName, BirthDate, IssueDate, ExpiryDate, IssueAuthority) 1NF
BirthDate
LicenseNum
DriverName
IssueDate
ExpiryDate
IssueAuthority
1
2
3
4
5
6
FD:
1. LicenseNum, DriverName → BirthDate
2. LicenseNum → IssueDate
3. LicenseNum → ExpiryDate
4. LicenseNum → IssueAuthority
5. IssueAuthority → IssueDate
6. IssueAuthority → ExpiryDate

$R_{1.1}$ (LicenseNum, DriverName, BirthDate)



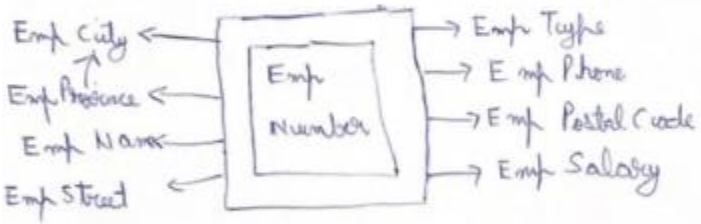
FD: { 1. LicenseNum, DriverName → BirthDate

$R_{1.2}$ (LicenseNum, IssueDate, ExpiryDate, IssueAuthority) 2NF



FD: { 1. LicenseNum → IssueDate, ExpiryDate, IssueAuthority
2. IssueAuthority → IssueDate, ExpiryDate

R= Full Time (Emp Number, Emp Type, Emp Phone, Emp Name, Emp City, Emp Street, Emp Province, Emp Postal Code, Emp Salary)



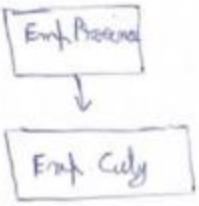
F.D: P.K(Emp Number) → Every Non Primary Key Attribute

Emp Province → Emp City.

R1.1) (Emp Number, Emp Type, Emp Phone, Emp Name, Emp City, Street, Emp Postal Code, Emp Salary)

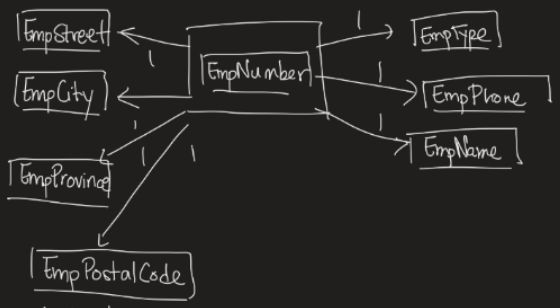
F.D: P.K(Emp Number) → Non Primary Key Attributes

R1.2) (Emp Province, Emp City)



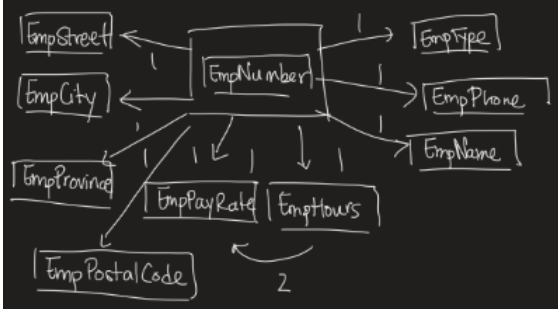
F.D: Emp Province → Emp City

R1.1 (EmpNumber, EmpType, EmpPhone, EmpName, EmpCity, EmpStreet, EmpProvince, EmpPostalCode)



FD: EmpNumber → EmpType, EmpPhone, EmpName, EmpCity, EmpStreet, EmpProvince, EmpPostalCode, EmpPayRate, EmpHours

R = PartTime (EmpNumber, EmpType, EmpPhone, EmpName, EmpCity, EmpStreet, EmpProvince, EmpPostalCode, EmpPayRate, EmpHours)

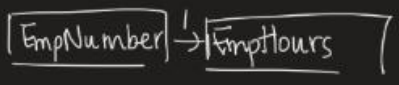


1NF
2NF

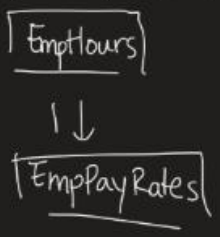
FD: 1. EmpNumber → EmpType, EmpPhone, EmpName, EmpCity, EmpStreet, EmpProvince, EmpPostalCode, EmpPayRate, EmpHours

2. EmpHours → EmpPayRate

R1.2.1 (EmpNumber, EmpHours)



R1.2.2 (EmpHours, EmpPayRate)

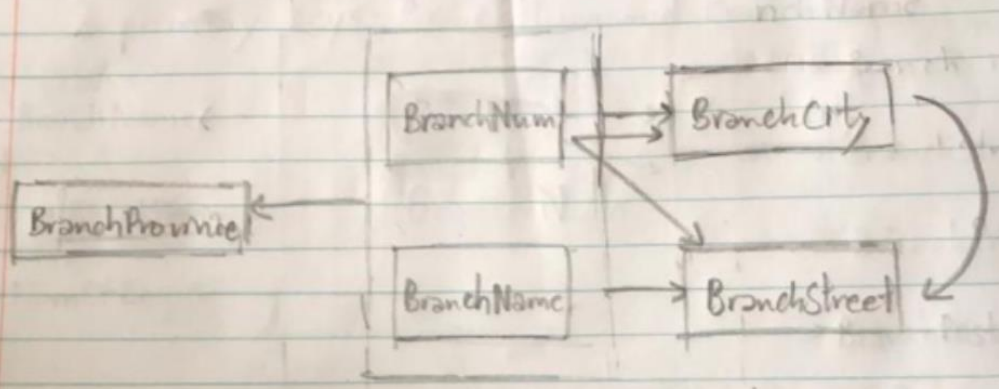


1NF
2NF
3NF

FD: 1. EmpNumber → EmpHours

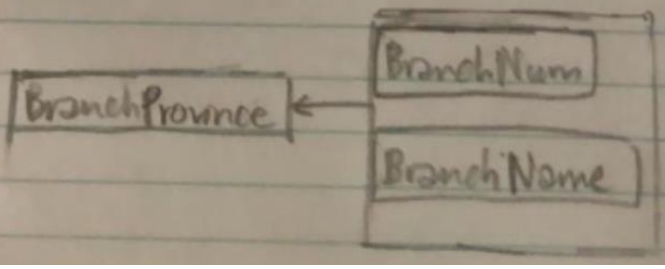
FD: 1. EmpHours → EmpPayRate

1NF R = RentalStoreBranch (BranchNum, BranchName, BranchPhone, NumOfEmp, BranchCity, BranchStreet, BranchProvince, BranchPostalCode)

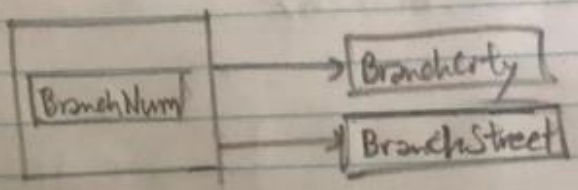


FD:
1. BranchNum, BranchName → BranchPhone, NumOfEmp, BranchCity, BranchStreet, BranchProvince, BranchPostalCode
2. BranchNum → BranchPhone
3. BranchNum → NumOfEmp
4. BranchNum → BranchCity
5. BranchNum → BranchStreet
6. BranchNum → BranchProvince
7. BranchNum → BranchPostalCode
8. BranchProvince → BranchPostalCode

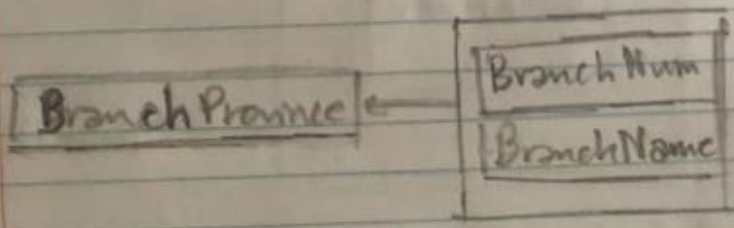
$R_{1.1}$ (BranchNum, BranchName, BranchProvince)



$R_{1.2}$ (BranchNum, BranchCity, BranchStreet) 2NF

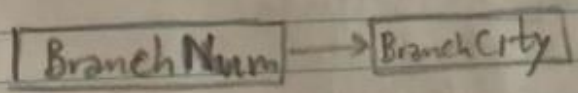


$R_{1.1}$ (BranchNum, BranchName, BranchProvince)



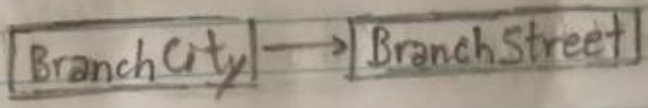
FD: BranchNum, BranchName → BranchProvince

$R_{1.2}$ (BranchNum, BranchCity)

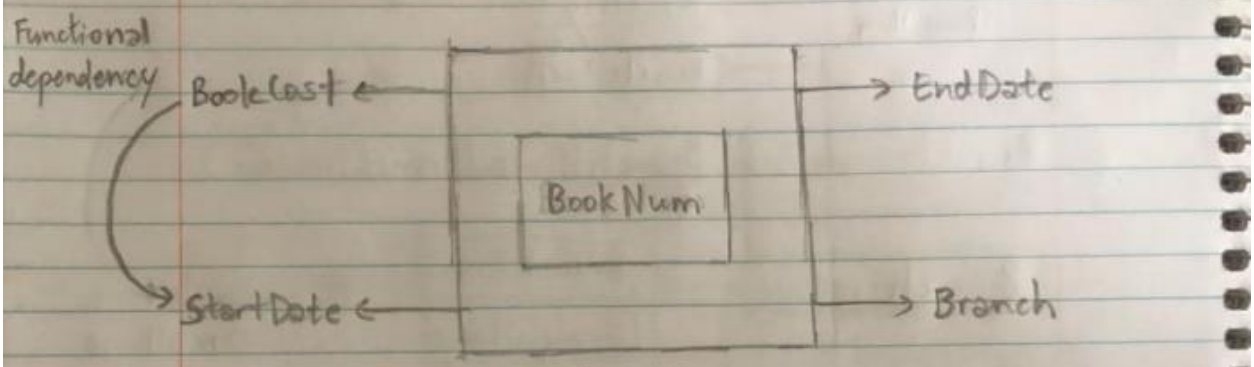


FD: BranchNum → BranchCity

$R_{1.3}$ (BranchCity, BranchStreet) 3NF

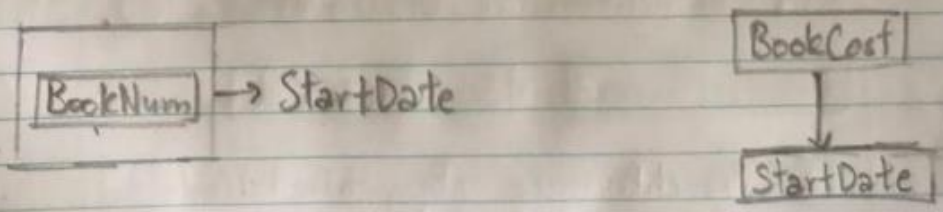


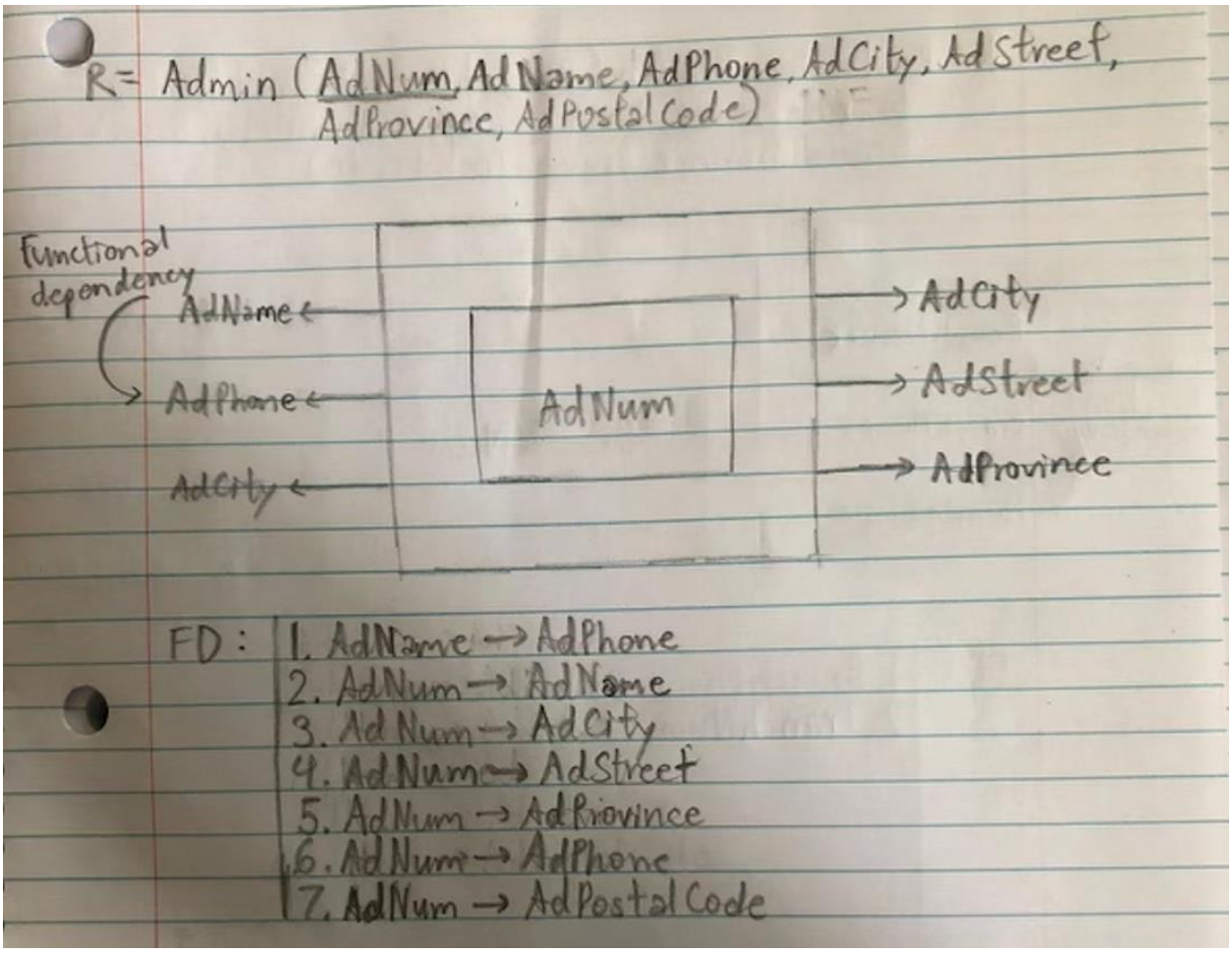
R = Booking (BookNum, BookCost, StartDate, EndDate, Branch)

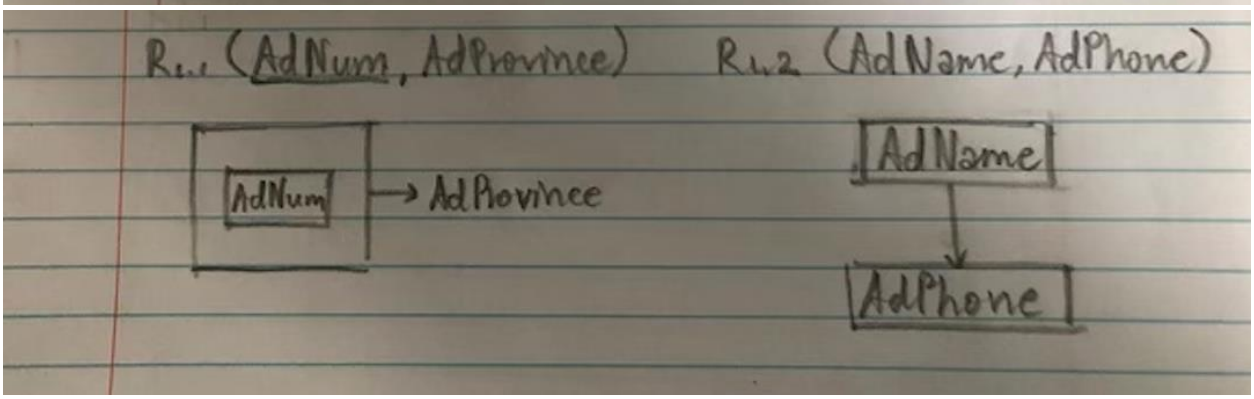


FD:
1. BookCost → StartDate
2. BookNum → BookCost
3. BookNum → StartDate
4. BookNum → EndDate
5. BookNum → Branch

$R_{1.1}$ (BookNum, StartDate) $R_{1.2}$ (BookCost, StartDate)







9. Normalization 3NF/BCNF using Bernstein's Algorithm and BCNF Algorithm

R = Customer (Cus Number, Cus Name, Cus Email, Cus Username, Cus Password, Cus Phone, Cus City, Cus Province, Cus Postal Code)

① FD:
- Cus Number → Cus Name (redundant)
- Cus Number → Cus Email
- Cus Number → Cus Username
- Cus Password → Cus Password
- Cus Number → Cus Phone
- Cus Number → Cus City
- Cus Number → Cus Province (redundant)
- Cus Number → Cus Postal Code
- Cus Email → Cus Name
- Cus City → Cus Province

② Checking for redundancy

for Cus Number → Cus Name

$(Cus\ Number)^+ = \{$ Cus email, Cus Name, Cus Username, Cus Password ... $\}$

As we still are able to find Cus Name, this F.D is redundant.

IIIly we find

Cus Number → Cus Postal code

IIIly we find

Cus Number → Cus Province to be redundant.

③ Finding Key.

Cus Number appears in LHS but not RHS but it is part of the key

$(\text{Cus Number})^+ = \{$ Cus Name, Cus Email, Cus Username, Cus Password, Cus Phone, Cus City, Cus Province, Cus Postal Code $\}$

Hence Cus Number is a key as it contains all the attributes

Now checking for Cus Email

$(\text{Cus Email}^+) = \{$ (Cus Name) $\}$, ∴ not a key as it doesn't contain all attributes

IIIly for Cus City

$(\text{Cus City}^+) = \{$ Cus Province $\}$ ∴ not a key as it does not contain all attributes

Hence Cus Number is the key.

(4) Convert to BCNF,

We find the FD

Cus Email → Cus Name and Cus City → Cus Province

is violating BCNF as neither Cus Email and Cus City are candidate key.

Hence we get.

R_1 (Cus Email, Cus Name)

R_2 (Cus City, Cus Province)

R_3 (Cus Number, Cus Username, Cus Password, Cus Phone, Cus Postal code)

R= Car((CarNumber, CarSeats, CarName, CarBrand, CarModel, CarYear, CarTransmission, CarClass, CarPlateNum, CarProvince, CarCity, CarStreet)

F.D: {
CarNumber → CarSeats
CarNumber → CarName
CarNumber → CarBrand
CarNumber → CarModel
CarNumber → CarYear → Redundant
CarNumber → CarTransmission
" → CarClass
" → CarPlate
" → CarProvince
" → CarCity
" → CarStreet
CarModel → CarYear.

Now we check for redundancy.

for CarNumber → CarYear

(CarNumber)$^+$ = {CarSeats, CarName, CarBrand, CarModel, CarTransmission, CarClass, ClassPlate, CarProvince, CarCity, CarStreet, CarYear}

As we still get CarYear we find

CarNumber → CarYear is a redundant F.D.

(3) Finding the key

As Car Number appears in LHS but not in RHS it is part of the key.

$(\text{Car Number})^+ = \{$ Car Seats, Car Name, Car Brand, Car Model, Car Transmission, Car Class, Car Plate, Car Province, Car City, Car Street, Car Year $\}$

is a key as it has all the attributes.

Now we try for

$(\text{Car Model})^+ = \{$ Car Year $\}$ not a key as it is missing key attributes.

Hence Car Number is key.

(4) Convert to BCNF.

Now we find the FD: Car Model → Car Year is violating BCNF as Car Model is not a Candidate key

Hence we break it into parts.

$R_1 = \{$ Car Number, Car Seats, Car Name, Car Brand, Car Transmission, Car Class, Car Plate, Car Province, Car City, Car Street. $\}$

$R_2 = \{$ Car Model, Car Year $\}$

R = Liability (Liab Number , Liab Type , Liab Expenses)

① FD :- { 1. Liab Number ⟶ Liab Type, Liab Expenses

② FD : { 1) Liab Number → Liab Type

2) Liab Number → Liab Expenses

Now Check for Redundancy

- Liab Number → Liab Type

  Liab Number$^+$ = { Liab Number, Liab Expenses }

- Liab Number → Liab Expenses

  Liab Number$^+$ = { Liab Number, Liab Type }

As the closure of has no overlap, there is no redundancy

③ R_1 (Liab Number, Liab Type, Liab Expenses) 3NF

④ As Liab Number is only in the LHS but not in the RHS, it is part of the key

hence Liab Number is the key

As Liab Number is the key and they system is already in 3NF

hence the table is in BCNF.

R = Rental (Rent Number, Rent Type, Rent Coverage, Rent Expenses, Term Length)

① F.D :

- Rent Number → Rent Type
- Rent Type → Rent Coverage
- ~~Rent Expenses~~ Rent Number → Rent Coverage
- Rent Number → Rent Coverage Rent Expenses
- Rent Number → Term Length
- ~~Rent Number → Rent Expenses~~

② Checking for Redundancy

for Rent Number → ~~Rent Coverage~~ Rent Coverage

we have $(\text{Rent Number})^+ = \{$ Rent Type, Rent coverage, Rent Expenses, Term Length $\}$

As we still find Rent Coverage, we conclude

Rent Number → Rent Coverge is redundant.

③ ~~Checking~~ Checking for keys,

As Rent Number is only on LHS but not on RHS, it is a key

Checking $(\text{rent Number})^+ = \{$ Rent Type, Rent Coverage, Rent Expenses, Term Length $\}$

As we find all the attributs

Rent Number is our key.

④ Convert it to BCNF

Ans Our table is not in BCNF as we have the F.D of

Rent Type → Rent Coverage

as Rent Type is not a key. To convert our table into BCNF we split our table into 2.

$R_1 = \{$ Rent Number, Rent Expenses, Term Length $\}$

$R_2 = \{$ Rent Type, Rent Coverage $\}$

R = Payment (PayNumber, PayTotal, PayType)

① FD: { 1. PayNumber → PayTotal, PayType

② FD: { 1. PayNumber → PayTotal
2. PayNumber → PayType

- PayNumber → PayTotal: PayNumber$^+$ = {PayNumber, PayType}
we do not get PayTotal, so not redundant
- PayNumber → PayType: PayNumber$^+$ = {PayNumber, PayTotal}
we do not get PayType, so not redundant
- all are fully dependent

③ R_1(PayNumber, PayTotal, PayType) 3NF

④ PayNumber$^+$ is a key

Since PayNumber is the primary key and the table is in 1NF, 2NF, and 3NF
Therefore the table is in BCNF.

R = R_1(PayNumber, PayTotal, PayType)

R = Bill (BillNumber, BillDate, BillProvince, BillCity, BillStreet, BillPostalCode)

① FD:
1. BillNumber → BillDate
2. BillNumber → BillProvince *
3. BillNumber → BillCity *
4. BillNumber → BillStreet *
5. BillNumber → BillPostalCode
6. BillPostalCode → BillProvince
7. BillPostalCode → BillStreet
8. BillPostalCode → BillCity

(2, 3, 4) redundant

BillNumber → BillCity: $(BillNumber)^+ = \{BillNumber, BillDate, BillPostalCode, BillCity, BillStreet, BillProvince\}$

we still get BillCity ⇒ redundant

BillNumber → BillStreet: $(BillNumber)^+ = \{BillNumber, BillDate, BillPostalCode, BillCity, BillStreet, BillProvince\}$

we still get BillStreet ⇒ redundant

BillNumber → BillProvince: $(BillNumber)^+ = \{BillNumber, BillDate, BillPostalCode, BillCity, BillStreet, BillProvince\}$

we still get BillProvince ⇒ redundant

②Remove Redundancies: FD:
1. BillNumber → BillDate
2. BillNumber → BillPostalCode
3. BillPostalCode → BillProvince
4. BillPostalCode → BillStreet
5. BillPostalCode → BillCity

They are in 2NF because all FDs are fully dependent.

③ BillNumber appears in LHS but not RHS (part of key)
BillProvince, BillStreet, BillCity, BillDate appear in RHS but not LHS (not part of key)

$(BillNumber)^+ = \{BillNumber, BillDate, BillPostalCode, BillProvince, BillStreet, BillCity\}$

is a key because you get all attributes

$(BillPostalCode)^+ = \{BillPostalCode, BillProvince, BillStreet, BillCity\}$ not a key

missing attributes (decompose)

$R_{11} = \{BillNumber, BillDate\}$

$R_{12} = \{BillPostalCode, BillStreet, BillCity, BillProvince\}$ BCNF

R = Discount (DiscNumber, DiscType, DiscTotal)

① FD: { 1. Disc Number → DiscTotal, DiscType

② FD: { 1. DiscNumber → Disc Total
2. DiscNumber → DiscType

- DiscNumber → DiscTotal : DiscNumber$^+$ = {DiscNumber, DiscType}
we do not get DiscTotal, so not redundant
- DiscNumber → DiscType : DiscNumber$^+$ = {DiscNumber, DiscTotal}
we do not get DiscType, so not redundant
- all are fully dependent

③ R_1 (DiscNumber, DiscTotal, DiscType) 3NF

④ DiscNumber + is a Key

Since DiscNumber is the primary key and the table is in 1NF, 2NF, and 3NF
Therefore the table is in BCNF.

R = R_1 (DiscNumber, DiscTotal, DiscType)

R = DriversLicense (LicenseNum, DriverName, BirthDate, IssueDate, ExpiryDate, IssueAuthority)
Bernstein Algorithm:

① FD:
1. LicenseNum, DriverName → BirthDate
2. LicenseNum → DriverName, IssueDate, ExpiryDate, IssueAuthority
3. IssueAuthority → IssueDate, ExpiryDate

② a) FD:
1. LicenseNum, DriverName → BirthDate
2. LicenseNum → DriverName
3. IssueAuthority → IssueDate
4. LicenseNum → IssueDate
5. LicenseNum → ExpiryDate
6. LicenseNum → IssueAuthority
7. IssueAuthority → ExpiryDate

- LicenseNum, DriverName → BirthDate : (LicenseNum, DriverName)$^+$ = {LicenseNum, DriverName, IssueDate, IssueAuthority, ExpiryDate}
we do not get BirthDate so not redundant

- LicenseNum → DriverName : (LicenseNum)$^+$ = {LicenseNum, IssueAuthority, IssueDate, ExpiryDate}
we do not get DriverName so not redundant

- IssueAuthority → IssueDate : (IssueAuthority)$^+$ = {IssueAuthority, ExpiryDate}
we do not get IssueDate so not redundant

LicenseNum → IssueDate : (LicenseNum)$^+$ = {DriverName, LicenseNum, IssueDate, IssueAuthority, ExpiryDate, BirthDate}
we got IssueDate, so this FD is redundant

LicenseNum → ExpiryDate : (LicenseNum)$^+$ = {DriverName, LicenseNum, IssueDate, IssueAuthority, ExpiryDate, BirthDate}
we got ExpiryDate, so this FD is redundant

LicenseNum → IssueAuthority : (LicenseNum)$^+$ = {LicenseNum, DriverNum, BirthDate, IssueDate, ExpiryDate}
we do not get IssueAuthority so not redundant

IssueAuthority → ExpiryDate : (IssueAuthority)$^+$ = {IssueAuthority, IssueDate}
we do not get ExpiryDate so not redundant

After removing redundancies:

FD:
1. LicenseNum, DriverName → BirthDate
2. LicenseNum → DriverName
3. IssueAuthority → IssueDate
5. LicenseNum → IssueAuthority
6. IssueAuthority → ExpiryDate

2b) - LicenseNum, DriverName → BirthDate: $(LicenseNum)^+ = \{LicenseNum, DriverName, IssueAuthority\}$

$(DriverName)^+ = \{DriverName\}$

Since we don't get BirthDate this FD is fully dependent

Since all FDs are fully dependent they are in 2NF.

③ IssueDate and ExpiryDate appear in RHS but LHS so they should not be part of a key.
All other attributes are on both LHS and RHS
LicenseNum does not appear in RHS but appears in LHS so it is part of a key.

- $LicenseNum^+ = \{LicenseNum, DriverName, IssueAuthority, IssueDate, ExpiryDate, BirthDate\}$

this is key because all attributes are found

④ FD:
1. LicenseNum, DriverName → BirthDate ⇒ R_1 (LicenseNum, DriverName, BirthDate)
2. LicenseNum → DriverName ⇒ R_2 (LicenseNum, DriverName)
3. IssueAuthority → IssueDate ⇒ R_3 (IssueAuthority, IssueDate)
5. LicenseNum → IssueAuthority ⇒ R_4 (LicenseNum, IssueAuthority)
6. IssueAuthority → ExpiryDate ⇒ R_5 (IssueAuthority, ExpiryDate)

R_1 (LicenseNum, DriverName, BirthDate)
R_2 (LicenseNum, DriverName)
R_3 (IssueAuthority, IssueDate)
R_4 (LicenseNum, IssueAuthority)
R_5 (IssueAuthority, ExpiryDate)

LicenseNum$^+$ is a key = {LicenseNum, DriverName, IssueAuthority, IssueDate, ExpiryDate, BirthDate}

IssueAuthority$^+$ is not a key = {IssueAuthority, ExpiryDate, IssueDate}

R_{11} = {LicenseNum, DriverName, BirthDate}

R_{12} = {IssueAuthority, IssueDate, ExpiryDate} BCNF

FD = {BookCost → StartDate, BookNum → BookCost,
BookNum → StartDate, BookNum → EndDate,
BookNum → Branch}

BookCost → StartDate : BookCost+ = {BookCost}

We do not get StartDate, so not redundant

BookNum → BookCost : BookNum+ = {BookNum, StartDate,
EndDate, Branch}

We do not get BookCost, so not redundant

BookNum → StartDate : BookNum+ = {BookNum, BookCost,
StartDate, EndDate,
Branch}

We get StartDate, so this FD is redundant

BookNum → EndDate : BookNum+ = {BookCost, StartDate,
Branch}

Not redundant

BookNum → Branch : BookNum+ = {BookCost, StartDate,
EndDate}

Not redundant

After removing redundancies, FD = {BookCost → StartDate,
BookNum → BookCost,
BookNum → EndDate,
BookNum → Branch}

BookCost → StartDate : R_1 (BookCost, StartDate)

BookNum → BookCost : R_2 (BookNum, BookCost)

BookNum → EndDate : R_3 (BookNum, EndDate)

BookNum → Branch : R_4 (BookNum, Branch)

BookNum is a key

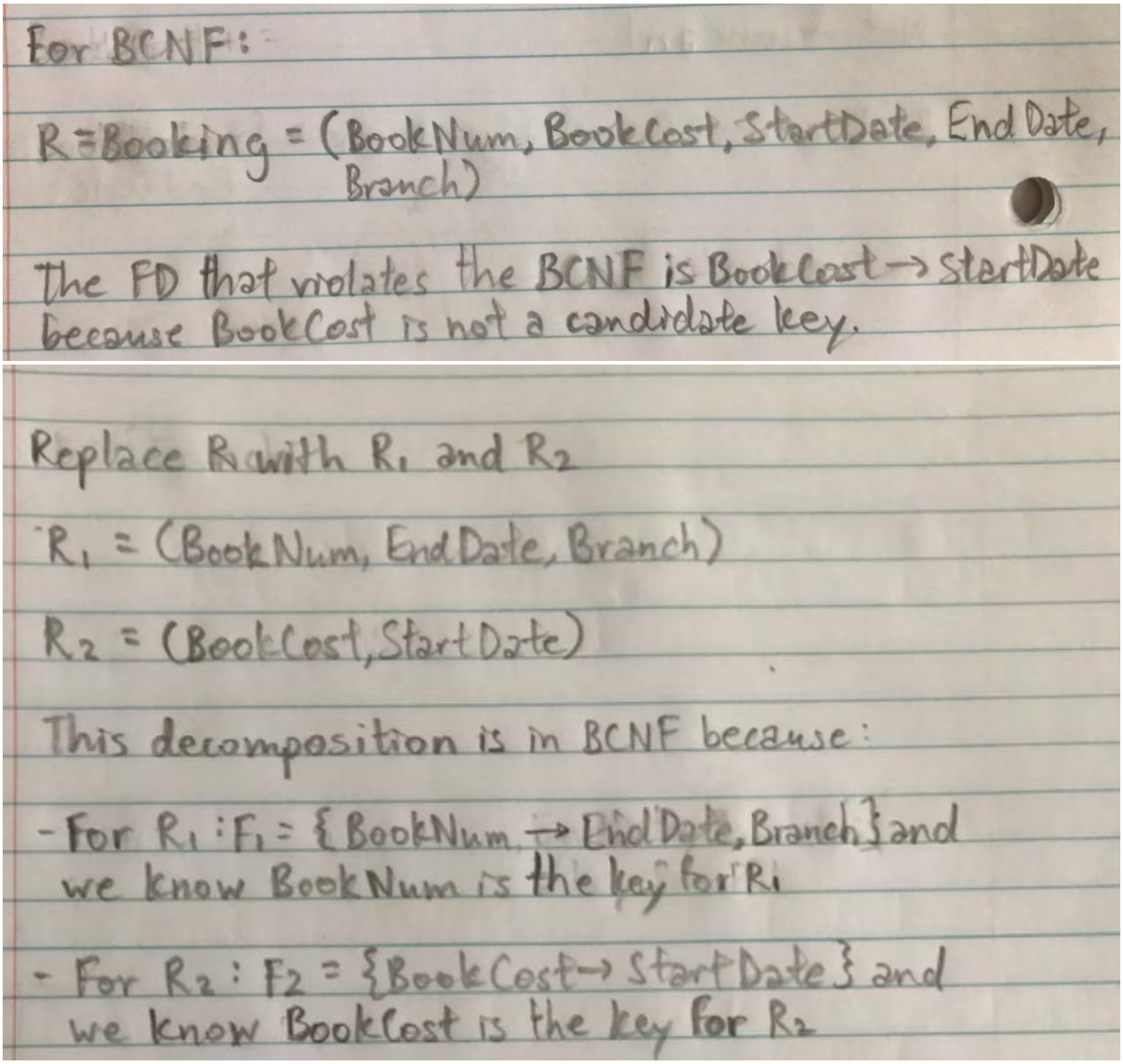
R_1 (BookCost, StartDate) with FD: BookCost → StartDate

R_2 (BookNum, BookCost) with FD: BookNum → BookCost

R_3 (BookNum, EndDate) with FD: BookNum → EndDate

R_4 (BookNum, Branch) with FD: BookNum → Branch

R_5 (BookNum) with no FD



10. Final Remarks

Overall, this assignment has been very informative and has allowed us to thoroughly understand databases and how they are created and used in the real world. Throughout the making of our project, we had to go back to previous parts of our assignments such as the ER diagram or queries and make adjustments to account for new concepts that we learn throughout the course and to make our overall project more efficient and errorless. We found the content that we learned in this course to be very helpful to create a database system that can be used for car rental dealerships. However, we were unable to finish the bonus GUI for our database due to time constraints, so our next steps would be to create a GUI for our UI implementation.

11. Project Demo

```
---Menu---
Please select one of the options:
1) Add a table
2) Drop a table
3) Insert data into a table
4) Update table
5) Select queries
6) Exit
```

Figure 1: Menu screen

```
---Menu---
Please select one of the options:
1) Add a table
2) Drop a table
3) Insert data into a table
4) Update table
5) Select queries
6) Exit
1
Created tables...
```

Figure 2: Option 1 selected, creating tables

```
---Menu---
Please select one of the options:
1) Add a table
2) Drop a table
3) Insert data into a table
4) Update table
5) Select queries
6) Exit
2
Deleted tables...
```

Figure 3: Option 2 selected, deleting tables

```
---Menu---
Please select one of the options:
1) Add a table
2) Drop a table
3) Insert data into a table
4) Update table
5) Select queries
6) Exit
3
Inserting records into the table...
Inserted data into tables...
```

Figure 4: Option 3 selected, inserting data into tables

```
---Menu---
Please select one of the options:
1) Add a table
2) Drop a table
3) Insert data into a table
4) Update table
5) Select queries
6) Exit
4
LiabNumber: 10, LiabType: Liability, LiabConverage: ttrg, LiabExpenses: 2001
LiabNumber: 110, LiabType: Liability, LiabConverage: ttrg, LiabExpenses: 30
LiabNumber: 1596, LiabType: Liability, LiabConverage: ttrg, LiabExpenses: 2030
```

Figure 5: Option 4 selecting, example of updating tables with printed new output

```
---Menu---
Please select one of the options:
1) Add a table
2) Drop a table
3) Insert data into a table
4) Update table
5) Select queries
6) Exit
5
Original data in table...
LiabNumber: 10, LiabType: Liability, LiabCoverage: ttrg, LiabExpenses: 2001
LiabNumber: 110, LiabType: Liability, LiabCoverage: ttrg, LiabExpenses: 30
LiabNumber: 1596, LiabType: Liability, LiabCoverage: ttrg, LiabExpenses: 2030
New data for query conditions...
LiabNumber: 110, LiabType: Liability, LiabCoverage: ttrg, LiabExpenses: 30
LiabNumber: 1596, LiabType: Liability, LiabCoverage: ttrg, LiabExpenses: 2030
```

Figure 6: Option 5 selected, example of a simple query for table and printing the original table as well as the table corresponding to the query

```
---Menu---
Please select one of the options:
1) Add a table
2) Drop a table
3) Insert data into a table
4) Update table
5) Select queries
6) Exit
6
Exiting menu...
```
Figure 6: Option 6 selected, exiting menu screen

12. Project Code (UI)

(a) <u>jdbc connection</u>

```
package jdbcoracleconnectiontemplate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.Properties;
import java.util.Scanner;

/**
 * This program makes database connection with Oracle on the Ryerson database
 *
 */
public class JdbcOracleConnectionTemplate {

   public static void main(String[] args) {

      Connection conn1 = null;

      try {

         Class.forName("oracle.jdbc.OracleDriver");
```

```
        String dbURL1 =
"jdbc:oracle:thin:username/password@oracle.scs.ryerson.ca:1521:orcl";

                        conn1 = DriverManager.getConnection(dbURL1);
        if (conn1 != null) {
           System.out.println("Connected with connection #1");
        }


           UImenu classObj = new UImenu(conn1);
           classObj.start(conn1);

      } catch (ClassNotFoundException ex) {
         ex.printStackTrace();
      } catch (SQLException ex) {
         ex.printStackTrace();
      } finally {
         try {
            if (conn1 != null && !conn1.isClosed()) {
               conn1.close();
            }

         } catch (SQLException ex) {
            ex.printStackTrace();
         }
      }
   }
}
```

(b) <u>UI menu</u>

```
package jdbcoracleconnectiontemplate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.ResultSet;
```

```
public class UImenu {

   private Connection conn;
   static final String QUERY1 = "SELECT LiabNumber, LiabType, LiabCoverage,
LiabExpenses FROM Liability";
   static final String QUERY = "SELECT LiabNumber, LiabType, LiabCoverage, LiabExpenses
FROM Liability";


   UImenu(Connection conn1) {
      this.conn = conn;
   }

   public void start(Connection conn1){

      System.out.println(" ");
      System.out.println("---Menu---");
      System.out.println("Please select one of the options:");
      System.out.println("1) Add a table");
      System.out.println("2) Drop a table");
      System.out.println("3) Insert data into a table");
      System.out.println("4) Update table");
      System.out.println("5) Select queries");
      System.out.println("6) Exit");

      Scanner myObj = new Scanner(System.in);
      String res = myObj.nextLine();

      switch (res){
      case "1":
            createtable(conn1);
            break;
      case "2":
            droptable(conn1);
            break;
      case "3":
            insertrecords(conn1);
            break;
      case "4":
            updaterecords(conn1);
```

```
            break;
        case "5":
            selectrecords(conn1);
            break;
        case "6":
           System.out.println("Exiting menu...");
           System.exit(0);
        }
    }

public void createtable(Connection conn1) {
  try (Statement stmt = conn1.createStatement()){
     String sql = "CREATE TABLE CUSTOMER " +
              "(CusNumber INTEGER not NULL, " +
              " CusName VARCHAR(255) not NULL, " +
              " CusEmail VARCHAR(255) unique not NULL, " +
              " CusUsername VARCHAR(255) unique not NULL, " +
              " CusPassword VARCHAR(255) not NULL, " +
              " CusPhone VARCHAR(255) not NULL, " +
              " CusCity VARCHAR(255) not NULL, " +
              " CusStreet VARCHAR(255) not NULL, " +
              " CusProvince VARCHAR(255) not NULL, " +
              " CusPostalCode VARCHAR(255) not NULL, " +
              " PRIMARY KEY ( CusNumber ))";

     String sql2 = "CREATE TABLE LIABILITY " +
              "(LiabNumber INTEGER not NULL, " +
              " LiabType VARCHAR(255) not NULL, " +
              " LiabCoverage VARCHAR(255) not NULL, " +
              " LiabExpenses INTEGER not NULL, " +
              " PRIMARY KEY ( LiabNumber ))";
    stmt.executeUpdate(sql);
            stmt.executeUpdate(sql2);

    System.out.println("Created tables...");
  } catch (SQLException e) {
    e.printStackTrace();
  }
  finally {
     start(conn1);
```

```
  }
}

  public void droptable(Connection conn1) {
  try(Statement stmt = conn1.createStatement()){
       String sql = "DROP TABLE CUSTOMER";
       String sql2 = "DROP TABLE LIABILITY";
       stmt.executeUpdate(sql);
       stmt.executeUpdate(sql2);
       System.out.println("Deleted tables...");
  } catch (SQLException e) {
    e.printStackTrace();
  }
 finally {
     start(conn1);
  }
}

 public void insertrecords(Connection conn1) {
  try(Statement stmt = conn1.createStatement()){
    System.out.println("Inserting records into the table...");
    String sql = "INSERT INTO LIABILITY VALUES (10, 'Liability', 'ttrg', 2001)";
    stmt.executeUpdate(sql);
    sql = "INSERT INTO LIABILITY VALUES (110, 'Liability', 'ttrg', 2002)";
    stmt.executeUpdate(sql);
    sql = "INSERT INTO LIABILITY VALUES (1596, 'Liability', 'ttrg', 2030)";
    stmt.executeUpdate(sql);
    System.out.println("Inserted data into tables...");
  } catch (SQLException e) {
    e.printStackTrace();
  }
       finally {
     start(conn1);
  }
}

public void updaterecords(Connection conn1) {
  try(Statement stmt = conn1.createStatement()){
    String sql = "UPDATE LIABILITY " +
      "SET LiabExpenses = 30 WHERE LiabNumber = 110";
```

```
      stmt.executeUpdate(sql);
      ResultSet rs = stmt.executeQuery(QUERY1);

      while(rs.next()){
        System.out.print("LiabNumber: " + rs.getInt("LiabNumber"));
        System.out.print(", LiabType: " + rs.getString("LiabType"));
        System.out.print(", LiabConverage: " + rs.getString("LiabCoverage"));
        System.out.println(", LiabExpenses: " + rs.getInt("LiabExpenses"));

      }
      rs.close();
    } catch (SQLException e) {
      e.printStackTrace();
    }
        finally {
       start(conn1);
    }
  }

   public void selectrecords(Connection conn1) {

      try(Statement stmt = conn1.createStatement()){
        System.out.println("Original data in table...");
        ResultSet rs = stmt.executeQuery(QUERY);
      while(rs.next()){
        System.out.print("LiabNumber: " + rs.getInt("LiabNumber"));
        System.out.print(", LiabType: " + rs.getString("LiabType"));
        System.out.print(", LiabCoverage: " + rs.getString("LiabCoverage"));
        System.out.println(", LiabExpenses: " + rs.getInt("LiabExpenses"));
      }

      System.out.println("New data for query conditions...");
      String sql = "SELECT LiabNumber, LiabType, LiabCoverage, LiabExpenses FROM
Liability" +
        " WHERE LiabNumber >= 101 ";
      rs = stmt.executeQuery(sql);

      while(rs.next()){
        System.out.print("LiabNumber: " + rs.getInt("LiabNumber"));
        System.out.print(", LiabType: " + rs.getString("LiabType"));
```

```
        System.out.print(", LiabCoverage: " + rs.getString("LiabCoverage"));
        System.out.println(", LiabExpenses: " + rs.getInt("LiabExpenses"));         }
      rs.close();
    } catch (SQLException e) {
      e.printStackTrace();
    }
        finally {
       start(conn1);
    }
}
}
```