DEPARTMENT OF COMPUTING

COMP2350/6350 2021 S2 – ASSIGNMENT TWO (10%)

Draft Due: 11:55pm Tuesday 28 September 2021

Due: 11:55pm Friday 01 October 2021 (Week 8)

Database Design & Manipulation

Please Print Clearly In **CAPITALS**

Surname	JIA
First Name	RUNDE
Student ID	44434065
Signature	Jerry Jia

Student Code of Conduct

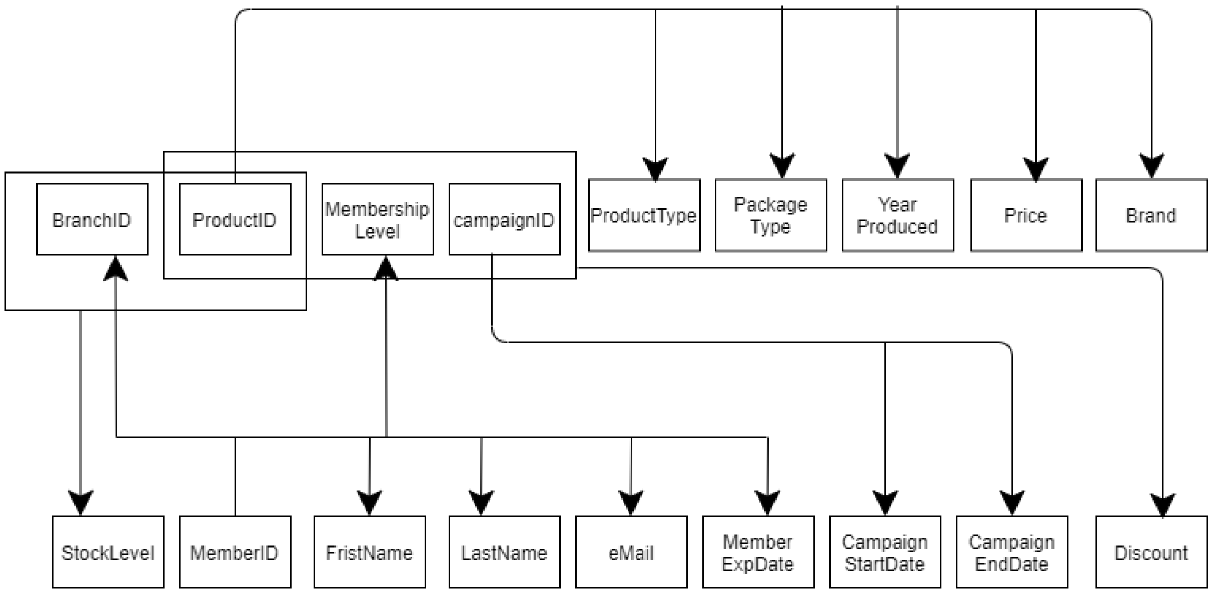
Macquarie University students have a responsibility to be familiar with the Student Code of Conduct: https://students.mq.edu.au/study/getting-started/student-conduct

Student Support

Task 1: Functional Dependencies

- *Identify the non-trivial FDs on the relation `Abnormal_Rel`. Supplement your description with diagram(s).*

ProductID -> ProductType, PackageType, YearProduced, Price, Brand
ProductID, BranchID -> StockLevel
campaignID -> CampaignStartDate, CampaignEndDate
MemberID -> FirstName, LastName, eMail, MembershipLevel, MemberExpDate, BranchID
ProductID, campaignID, MembershipLevel -> Discount



- *Identify the Candidate key(s) of `Abnormal_Rel`.*

(ProductID, BranchID, campaignID, MemberID)

Task 2: Anomalies

Note. *How you structure your answer is flexible. One possible structure is given below.*

- *Anomalies of Form_1*
    - *determine if the relation `Abnormal_Rel` is susceptible to it*
    - *Support your determination with adequate explanation and a small example (instance of the relation `Abnormal_Rel`).*

Relation Abnormal_Rel is susceptible to insertion anomaly. Insertion anomaly is that when we want to insert an attribute, it depends on other attributes. For example, we cannot insert a new branch that has no product stored yet.

- *Anomalies of Form_2*
    - *determine if the relation `Abnormal_Rel` is susceptible to it*
    - *Support your determination with adequate explanation and a small example (refer to the same or a different instance of the relation `Abnormal_Rel`).*

Relation Abnormal_Rel is susceptible to modification anomaly. We need to update any attribute at multiple locations instead of updating it directly at one place. For example, we need to update the price of one product, we need to change it in a number of places, risking inconsistency due to carelessness.

- *Anomalies of Form_3*
    - *determine if the relation `Abnormal_Rel` is susceptible to it*
    - *Support your determination with adequate explanation and a small example (refer to the same or a different instance of the relation `Abnormal_Rel`).*

Relation Abnormal_Rel is susceptible to deletion anomaly. It means if delete an attribute which is not needed it affects in deletion of the other depended attributes which are needed. For example, if one product is out of stock, we want to delete that information. But only one branch is storing that product. Then the whole branch information will be deleted.

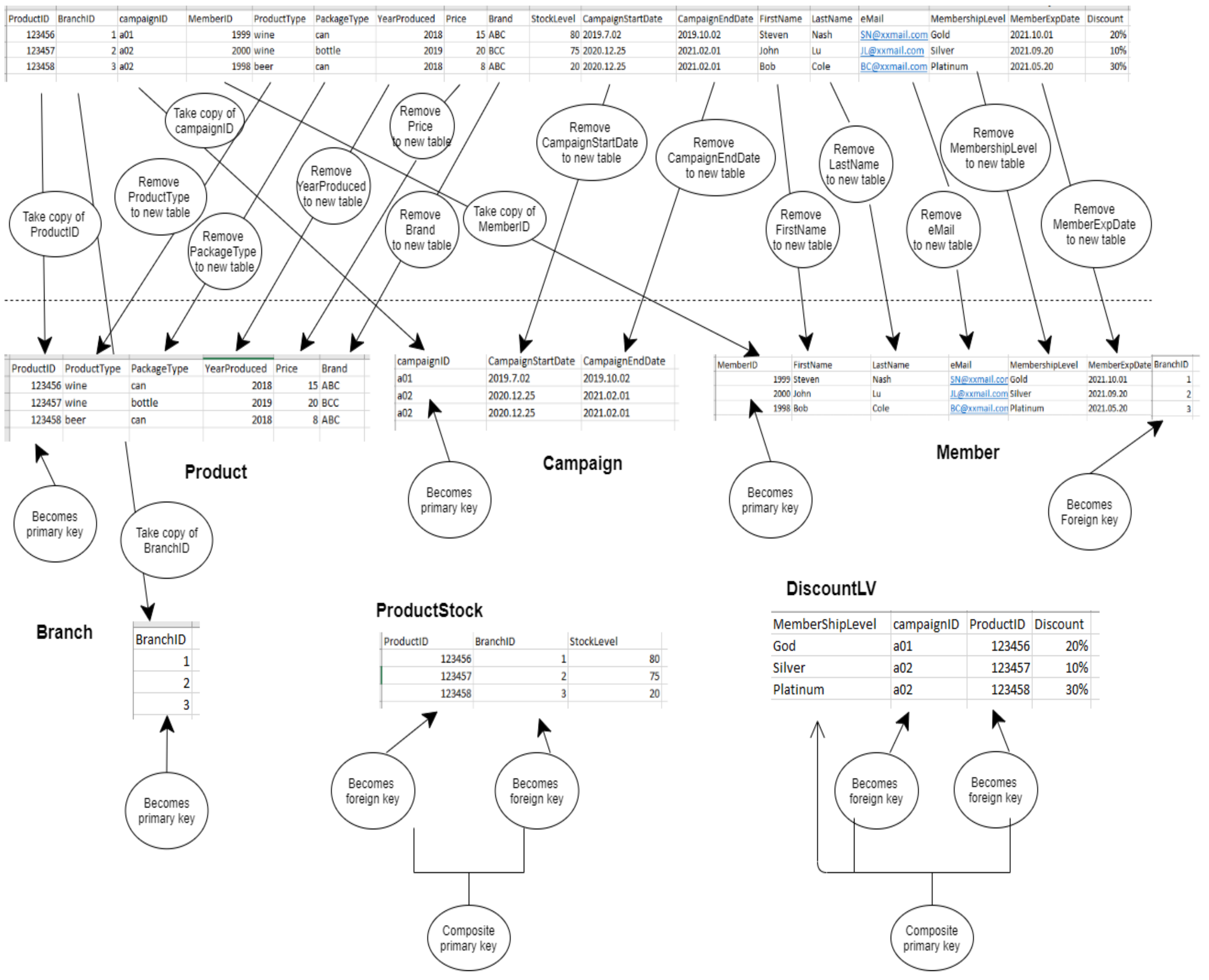
Task 3: Normalization

- *What is the highest NF that the relation `Abnormal_Rel` satisfies? Explain why.*

The relational schema is in FIRST NORMAL FORM, as there is no composite attribute, but there are partial and transitive dependency present in the relation. Several attributes that are dependent only on part of the key, thus violating 2NF.

Values in FirstName, LastName, eMail, MembershipLevel, MemberExpDate column can be worked out from only MemeberID, so table is only in 1NF.

- *Normalize/decompose `Abnormal_Rel` until you get relations that are in 3NF. Use appropriate illustration to aid the understanding of your work.*



Take copy of ProductID
Take copy of campaignID
Remove ProductType to new table
Remove PackageType to new table
Remove YearProduced to new table
Remove Price to new table
Remove Brand to new table
Take copy of MemberID
Remove CampaignStartDate to new table
Remove CampaignEndDate to new table
Remove FirstName to new table
Remove LastName to new table
Remove eMail to new table
Remove MembershipLevel to new table
Remove MemberExpDate to new table
Becomes primary key
Take copy of BranchID
Product
Becomes primary key
Campaign
Becomes primary key
Member
Becomes Foreign key
Branch
Becomes primary key
ProductStock
Becomes foreign key
Becomes foreign key
Composite primary key
DiscountLV
Becomes foreign key
Becomes foreign key
Composite primary key

ProductID	BranchID	campaignID	MemberID	ProductType	PackageType	YearProduced	Price	Brand	StockLevel	CampaignStartDate	CampaignEndDate	FirstName	LastName	eMail	MembershipLevel	MemberExpDate	Discount
123456	1	a01	1999	wine	can	2018	15	ABC	80	2019.7.02	2019.10.02	Steven	Nash	SN@xxmail.com	Gold	2021.10.01	20%
123457	2	a02	2000	wine	bottle	2019	20	BCC	75	2020.12.25	2021.02.01	John	Lu	JL@xxmail.com	Silver	2021.09.20	10%
123458	3	a02	1998	beer	can	2018	8	ABC	20	2020.12.25	2021.02.01	Bob	Cole	BC@xxmail.com	Platinum	2021.05.20	30%

Product

ProductID	ProductType	PackageType	YearProduced	Price	Brand
123456	wine	can	2018	15	ABC
123457	wine	bottle	2019	20	BCC
123458	beer	can	2018	8	ABC

Campaign

campaignID	CampaignStartDate	CampaignEndDate
a01	2019.7.02	2019.10.02
a02	2020.12.25	2021.02.01
a02	2020.12.25	2021.02.01

Member

MemberID	FirstName	LastName	eMail	MembershipLevel	MemberExpDate	BranchID
1999	Steven	Nash	SN@xxmail.con	Gold	2021.10.01	1
2000	John	Lu	JL@xxmail.com	Silver	2021.09.20	2
1998	Bob	Cole	BC@xxmail.con	Platinum	2021.05.20	3

Branch

BranchID
1
2
3

ProductStock

ProductID	BranchID	StockLevel
123456	1	80
123457	2	75
123458	3	20

DiscountLV

MemberShipLevel	campaignID	ProductID	Discount
God	a01	123456	20%
Silver	a02	123457	10%
Platinum	a02	123458	30%

Product (ProductID, ProductType, PackageType, YearProduced, Price, Brand)

Branch (BranchID)

Campaign (campaignID, CampaignStartDate, CampaignEndDate)

Member (MemberID, FirstName, LastName, eMail, MembershipLevel, MemeberExpDate, BranchID)

ProductStock (ProductID, BranchID, StockLevel)

DiscountLV (MembershipLevel, campaignID, ProductID, Discount)

Product Primary Key: ProductID;

Branch Primary Key: BranchID;

Campaign Composite Primary Key: campaignID;

Member Primary Key: MemberID;

ProductStock Composite Primary Key: (ProductID(Foreign Key), BranchID(Foreign Key))

DiscountLV Composite Primary Key: (MembershipLevel, campaignID (Foreign Key), ProductID(Foreign Key))

After removed partial and transitive dependency present in the relation, the relation now is in 3NF. All attributes are dependent on the whole primary key.

- *Check if the resultant relations are in BCNF. If not, decompose them as necessary until you get all of them in BCNF.*

The primary key is the only determinant (and the only candidate key), so by definition this is in BCNF.

Task 4: Table Creation and Population

- *Copy and paste your DDL code for creating each table/relation in BCNF obtained in Task 3.*

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';



drop table if exists Product, Branch, Campaign, Member, ProductStock, DiscountLV;


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

-- Table `Product`

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

CREATE TABLE IF NOT EXISTS `Product` (

  `ProductID` VARCHAR(10) NOT NULL,

  `ProductType` VARCHAR(45) NULL,

  `PackageType` VARCHAR(45) NULL,

  `YearProduced` INT NULL,

  `Price` DOUBLE NULL,

  `Brand` VARCHAR(45) NULL,

  PRIMARY KEY (`ProductID`))

ENGINE = InnoDB;



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

-- Table `Campaign`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Campaign` (
  `campaignID` VARCHAR(10) NOT NULL,
  `CampaignStartDate` DATE NULL,
  `CampaignEndDate` DATE NULL,
  PRIMARY KEY (`campaignID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Branch`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Branch` (
  `BranchID` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`BranchID`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `Member`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Member` (
  `MemberID` INT NOT NULL,
  `FirstName` VARCHAR(45) NULL,
  `LastName` VARCHAR(45) NULL,
  `eMail` VARCHAR(45) NULL,
  `MembershipLevel` VARCHAR(45) NULL,
  `MemberExpDate` DATE NULL,
 `BranchID` VARCHAR(10) NOT NULL,
 PRIMARY KEY (`MemberID`),
  INDEX `fk_Branch_has_Member_idx` (`BranchID` ASC),
 CONSTRAINT `fk_Branch_has_Member`
   FOREIGN KEY (`BranchID`)
   REFERENCES `Branch` (`BranchID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `ProductStock`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ProductStock` (
 `ProductID` VARCHAR(10) NOT NULL,
 `BranchID` VARCHAR(10) NOT NULL,
 `StockLevel` INT NULL,
 PRIMARY KEY (`BranchID`, `ProductID`),
 INDEX `fk_Branch_has_Product_idx` (`ProductID` ASC),
 CONSTRAINT `fk_Branch_has_Product`
   FOREIGN KEY (`ProductID`)
   REFERENCES `Product` (`ProductID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
 INDEX `fk_Branch_has_Product_idx2` (`BranchID` ASC),
 CONSTRAINT `fk_Branch_has_Product2`
```

```
    FOREIGN KEY (`BranchID`)
    REFERENCES `Branch` (`BranchID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `DiscountLV`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `DiscountLV` (
  `MembershipLevel` VARCHAR(45) NOT NULL,
  `campaignID` VARCHAR(10) NOT NULL,
  `ProductID` VARCHAR(10) NOT NULL,
  `Discount` VARCHAR(10) NULL,
  PRIMARY KEY (`MembershipLevel`, `campaignID`, `ProductID`),
  INDEX `fk_campaignID_idx` (`campaignID` ASC),
  CONSTRAINT `fk_campaignID`
    FOREIGN KEY (`campaignID`)
    REFERENCES `Campaign` (`campaignID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
INDEX `fk_ProductID_idx` (`ProductID` ASC),
  CONSTRAINT `fk_ProductID`
    FOREIGN KEY (`ProductID`)
    REFERENCES `Product` (`ProductID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
    ENGINE = InnoDB;



SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

- *Copy and paste your SQL code for inserting at least five rows of data into each of these table.*

```
-- --------------------------------------
-- Insert into 'Product'
-- --------------------------------------
INSERT INTO Product values ('12345', 'wine', 'bottle', '2010', '25.00', 'Penfold Grange');
INSERT INTO Product values ('12346', 'beer', 'can', '2019', '8.50', 'Corona Extra');
INSERT INTO Product values ('12347', 'beer', 'can', '2018', '10.50', 'Victoria Bitter');
INSERT INTO Product values ('12348', 'spirit', 'bottle', '2008', '50.00', 'Absolut Vodaka');
INSERT INTO Product values ('12349', 'wine', 'bottle', '2008', '30.00', 'Penfold Grange');


-- --------------------------------------
-- Insert into 'Branch'
-- --------------------------------------
INSERT INTO Branch values ('1');
INSERT INTO Branch values ('2');
INSERT INTO Branch values ('3');
INSERT INTO Branch values ('4');
INSERT INTO Branch values ('5');
```

```
-- -------------------------------------

-- Insert into 'Campaign'

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

INSERT INTO Campaign values ('a01', '2021-12-10', '2021-12-31');

INSERT INTO Campaign values ('a02', '2021-01-01', '2021-05-31');

INSERT INTO Campaign values ('a03', '2019-05-05', '2019-07-05');

INSERT INTO Campaign values ('a04', '2022-12-10', '2022-12-31');

INSERT INTO Campaign values ('a05', '2020-09-30', '2020-10-10');


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

-- Insert into 'Member'

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

INSERT INTO Member values ('1999', 'Nash', 'Steve', 'nashs@xx.mail', 'Gold', '2021-11-05',
'1');

INSERT INTO Member values ('1998', 'Simone', 'Singh', 'simones@xx.mail', 'Gold', '2021-11-
20','2');

INSERT INTO Member values ('2000', 'Kiki', 'West', 'kikiw@xx.mail', 'Silver', '2021-12-
20','3');

INSERT INTO Member values ('1987', 'Dickson', 'Wu', 'dicksonw@xx.mail', 'Platitum', '2021-
12-30','1');

INSERT INTO Member values ('2001', 'Tom', 'Jerry', 'tomj@xx.mail', 'Silver', '2022-01-
05','1');

INSERT INTO Member values ('1982', 'Lara', 'Howard', 'larah@xx.mail', 'Gold', '2023-11-
05','4');

INSERT INTO Member values ('2015', 'Fisher', 'Derrick', 'fisherd@xx.mail', 'Platitum', '2022-
05-05','5');


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

-- Insert into 'ProductStock'

-- -------------------------------------
```

INSERT INTO ProductStock values ('12345', '1', '10');

INSERT INTO ProductStock values ('12345', '2', '30');

INSERT INTO ProductStock values ('12347', '1', '8');

INSERT INTO ProductStock values ('12348', '3', '100');

INSERT INTO ProductStock values ('12349', '2', '1');

INSERT INTO ProductStock values ('12346', '4', '10');

INSERT INTO ProductStock values ('12346', '5', '50');

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

-- *Insert into 'DiscountLV'*

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

INSERT INTO DiscountLV values ('Gold', 'a01', '12346', '20%');

INSERT INTO DiscountLV values ('Gold', 'a01', '12347', '20%');

INSERT INTO DiscountLV values ('Platitum', 'a01', '12348', '40%');

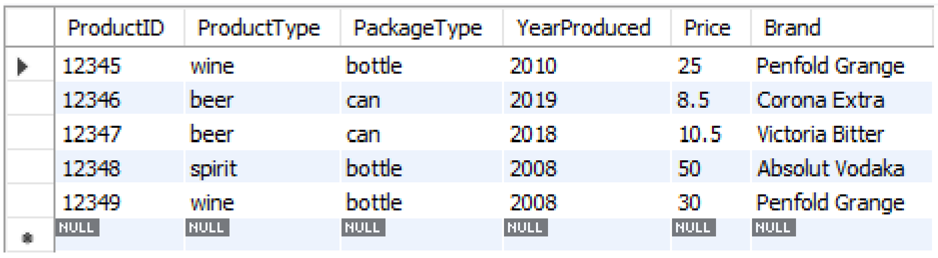
INSERT INTO DiscountLV values ('Platitum', 'a02', '12348', '35%');

INSERT INTO DiscountLV values ('Silver', 'a01', '12346', '10%');

INSERT INTO DiscountLV values ('Silver', 'a05', '12345', '10%');

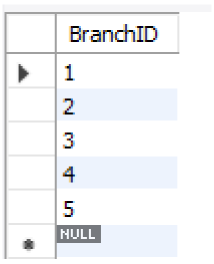
- *Copy and paste the SELECT * query to display the content of each table above, and screenshot of the content as displayed.*

SELECT * FROM Product;



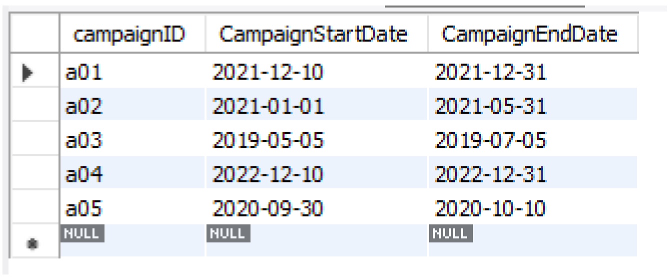
ProductID	ProductType	PackageType	YearProduced	Price	Brand
12345	wine	bottle	2010	25	Penfold Grange
12346	beer	can	2019	8.5	Corona Extra
12347	beer	can	2018	10.5	Victoria Bitter
12348	spirit	bottle	2008	50	Absolut Vodaka
12349	wine	bottle	2008	30	Penfold Grange
NULL	NULL	NULL	NULL	NULL	NULL

SELECT * FROM Branch;



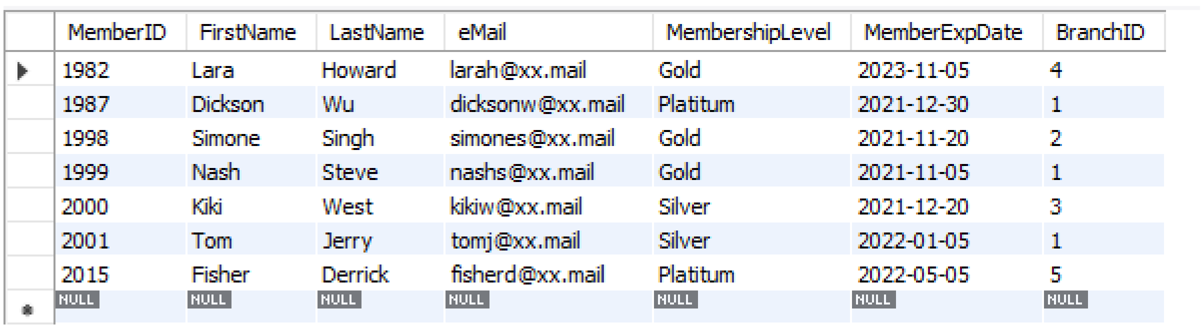
BranchID
1
2
3
4
5
NULL

SELECT * FROM Campaign;



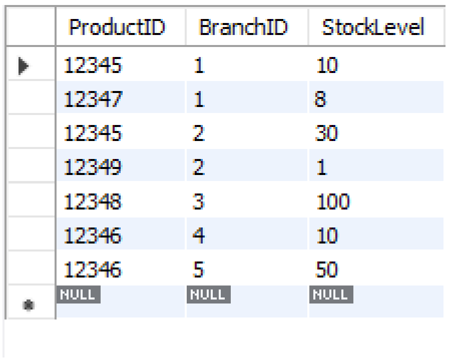
campaignID	CampaignStartDate	CampaignEndDate
a01	2021-12-10	2021-12-31
a02	2021-01-01	2021-05-31
a03	2019-05-05	2019-07-05
a04	2022-12-10	2022-12-31
a05	2020-09-30	2020-10-10
NULL	NULL	NULL

SELECT * FROM Member;



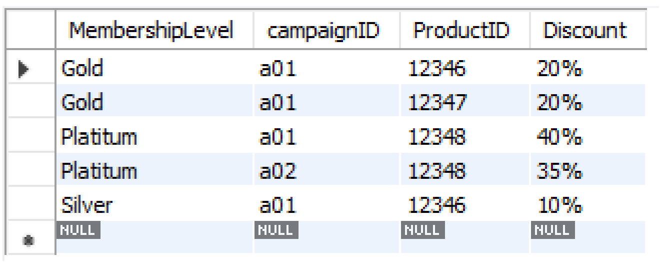
MemberID	FirstName	LastName	eMail	MembershipLevel	MemberExpDate	BranchID
1982	Lara	Howard	larah@xx.mail	Gold	2023-11-05	4
1987	Dickson	Wu	dicksonw@xx.mail	Platitum	2021-12-30	1
1998	Simone	Singh	simones@xx.mail	Gold	2021-11-20	2
1999	Nash	Steve	nashs@xx.mail	Gold	2021-11-05	1
2000	Kiki	West	kikiw@xx.mail	Silver	2021-12-20	3
2001	Tom	Jerry	tomj@xx.mail	Silver	2022-01-05	1
2015	Fisher	Derrick	fisherd@xx.mail	Platitum	2022-05-05	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SELECT * FROM ProductStock;



ProductID	BranchID	StockLevel
12345	1	10
12347	1	8
12345	2	30
12349	2	1
12348	3	100
12346	4	10
12346	5	50
NULL	NULL	NULL

SELECT * FROM DiscountLV;



MembershipLevel	campaignID	ProductID	Discount
Gold	a01	12346	20%
Gold	a01	12347	20%
Platitum	a01	12348	40%
Platitum	a02	12348	35%
Silver	a01	12346	10%
NULL	NULL	NULL	NULL

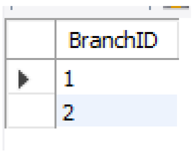
Task 5: SQL Queries

Copy and paste the SQL queries followed by their results (screenshot) for each of the following query

[Query 1] *List the branches (ID) of MA that have in stock at least 5 bottles of Penfold Grange 2010.*

```
        SELECT BranchID
        FROM ProductStock
        WHERE ProductID = (
                SELECT ProductID
          FROM Product
          WHERE PackageType = 'bottle'
          AND Brand = 'Penfold Grange'
          AND YearProduced = '2010'
) AND StockLevel >= 5;
```

Result:

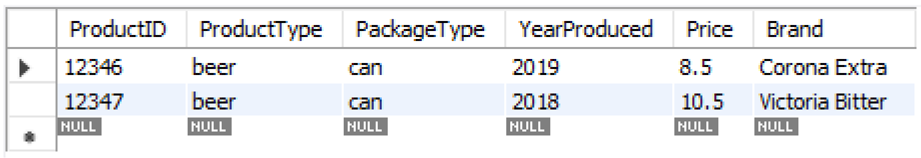


	BranchID
▶	1
	2

[Query 2] *List details of each beer that Simone Singh will be entitled to get 20% discount on.*

```
        SELECT *
        FROM Product
        WHERE ProductID IN (
                SELECT ProductID
          FROM DiscountLV
          WHERE Discount = '20%'
          AND MembershipLevel = (
                        SELECT MembershipLevel
            FROM Member
            WHERE FirstName =  'Simone'
            AND LastName = 'Singh'
          )
          AND campaignID = (
                        SELECT campaignID
            FROM Campaign
        WHERE '2021-12-24' BETWEEN CampaignStartDate AND CampaignEndDate
      )
);
```

Result:

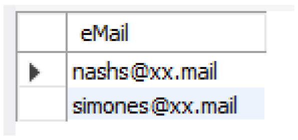


	ProductID	ProductType	PackageType	YearProduced	Price	Brand
▶	12346	beer	can	2019	8.5	Corona Extra
	12347	beer	can	2018	10.5	Victoria Bitter
*	NULL	NULL	NULL	NULL	NULL	NULL

[Query 3] *Generate a list of all email addresses of members whose card will expire in the month after the coming month, ordered appropriately.*

```
    SELECT eMail
    FROM Member
    WHERE MemberExpDate BETWEEN DATE_SUB(
      LAST_DAY(
        DATE_ADD(NOW(), INTERVAL 2 MONTH)
      ),
      INTERVAL DAY(
        LAST_DAY(
          DATE_ADD(NOW(), INTERVAL 2 MONTH)
        )
      )-1 DAY) AND LAST_DAY(
      DATE_ADD(NOW(), INTERVAL 2 MONTH)
    )
    ORDER BY BranchID ASC,
    MemberExpDate ASC,
eMail ASC;
```

Result:



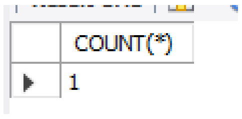
	eMail
▶	nashs@xx.mail
	simones@xx.mail

[Query 4] *Determine how many times Penfold Grange 2010 has gone on sale since Covid-19 related lockdown started (assume it to be March 01, 2020).*

```
    SELECT COUNT(*)
    FROM DiscountLV
    WHERE ProductID = (
            SELECT ProductID
      FROM Product
```

```
		WHERE Brand = 'Penfold Grange' AND YearProduced = '2010'
	)
	AND campaignID in (
			SELECT campaignID
		FROM Campaign
		WHERE (CampaignEndDate BETWEEN '2020-03-01' AND curdate())
		OR (CampaignStartDate BETWEEN '2020-03-01' AND curdate())
);
```

Result:



```
COUNT(*)
1
```