Gym Chain Database Design

INFO 6210
Data Management and Database Design

Team 12: Yuwei Cao
Jian Xiao
Jiaqi Wang
Jingya Zhou
Shuai Shao

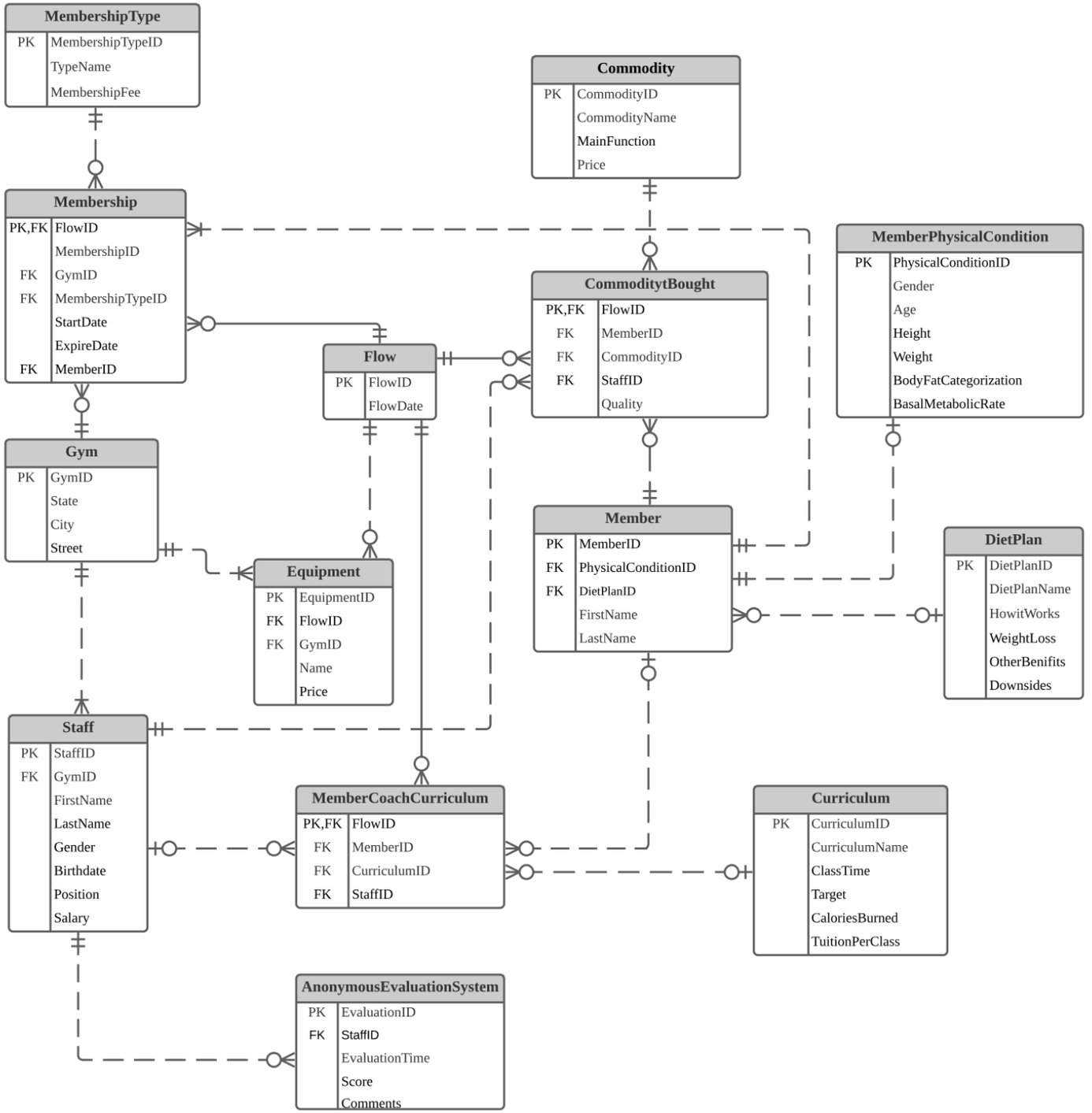
Database Purpose

The purpose of the database is maintaining and updating the basic data of staff, member, curriculum and flow in a gym chain to make daily affairs more smoothly and effectively while providing some suggestions for future development. It will be used by staffs and members under different permissions.

BUSINESS RULES

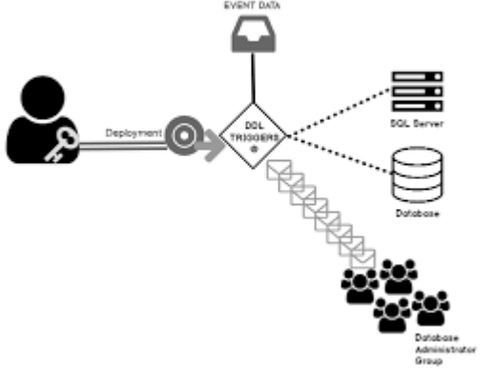
- Each staff works at exactly one gym.
- Each gym has one or more staffs.
- Each member has one membership.
- Each gym has zero or more members.
- Each membership belongs to one membership type.
- Each membership type has zero or more members.
- Each gym has one or more equipment.
- Each equipment was bought by one gym.
- Each staff has one position.
- Each position has zero or more staffs.
- Each staff has zero or more evaluations.
- Each evaluation record belongs to one staff.
- Each member can be taught by zero or more coaches.
- Each member can attend zero or more curriculums.
- Each curriculum has zero or more members.
- Each curriculum is conducted by zero or more coaches.
- Each coach can conduct zero or more curriculums.
- Each member can have zero or one diet plan.
- Each diet plan is used by zero or more members.
- Each member has zero or one physical condition research.
- Each physical condition record belongs to one member.
- Each member may buy zero or more commodities.
- Each member may buy commodities via zero or more staffs.
- Each commodity may be bought by zero or more members.
- Each commodity may be sold by zero or more staffs several times.
- Each staff may sell zero or more commodities.
- Each staff may sell commodities to zero or more members.
- Each time of applying a membership generates one flow record.
- Each equipment procurement generates one flow record.
- Each transaction of commodity has one flow record.
- Each signing up for a curriculum has one flow record.

Entity Relationship Diagram



DDL and Data Insertion

All tables were created with primary and foreign key relationships based on ERD. Sample data has been inserted in all of them.




```sql
CREATE TABLE MembershipType
(MembershipTypeID VARCHAR(10) PRIMARY KEY,
 TypeName VARCHAR(30),
 MembershipFee MONEY NOT NULL
 );
 CREATE TABLE Gym
(GymID VARCHAR(10) PRIMARY KEY,
 OpeningDate Date,
 State VARCHAR(50) NOT NULL,
 City VARCHAR(50) NOT NULL,
 Street VARCHAR(50) NOT NULL
);
 CREATE TABLE Staff
(StaffID VARCHAR(10) PRIMARY KEY,
 GymID VARCHAR(10) REFERENCES Gym(GymID),
 FirstName VARCHAR(50) NOT NULL,
 LastName VARCHAR(50) NOT NULL,
 Gender VARCHAR(5) CHECK(Gender in ('M', 'F')) NOT NULL,
 BirthDate DATE,
 Age AS DATEDIFF(hour,BirthDate,GETDATE())/8766,
 Position VARCHAR(50) NOT NULL,
 RankLevel INT CHECK(RankLevel between 1 and 5) NOT NULL,
 Salary_Month MONEY NOT NULL
);
CREATE TABLE AnonymousEvaluationSystem
(EvaluationID VARCHAR(10) PRIMARY KEY,
 StaffID VARCHAR(10) REFERENCES Staff(StaffID),
 EvaluationTime DATE DEFAULT GETDATE(),
 Score INT CHECK(Score between 1 and 5) NOT NULL,
 Comments VARCHAR(100) DEFAULT 'None'
);
  CREATE TABLE Commodity
(CommodityID VARCHAR(10) PRIMARY KEY,
CommodityName VARCHAR(50),
MainFunction VARCHAR(100),
Price MONEY
);
```

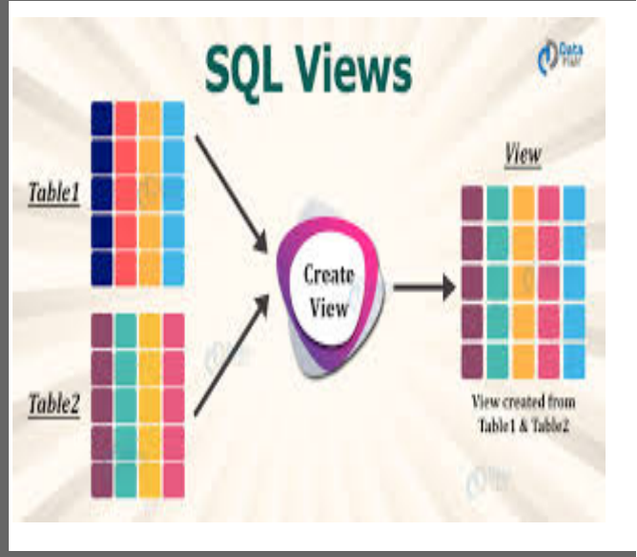
```sql
CREATE TABLE MemberPhysicalCondition
(PhysicalConditionID VARCHAR(10) PRIMARY KEY,
Gender VARCHAR(5) CHECK(Gender IN ('M','F')) NOT NULL,
DateOfBirth Date,
Age AS DATEDIFF(HOUR, DateOfBirth, GETDATE())/8766,
Height_CM FLOAT,
Weight_KG FLOAT,
BodyFatPercentage FLOAT
);
CREATE TABLE Member
(MemberID VARCHAR(10) PRIMARY KEY NOT NULL,
PhysicalConditionID VARCHAR(10) REFERENCES MemberPhysicalCondition(PhysicalConditionID),
DietPlanID VARCHAR(10) REFERENCES DietPlan(DietPlanID),
FirstName VARCHAR(50),
LastName VARCHAR(50) NOT NULL
);
CREATE TABLE Membership
(MembershipID VARCHAR(10) PRIMARY KEY,
 FlowID VARCHAR(10) UNIQUE NOT NULL REFERENCES Flow(FlowID),
 GymID VARCHAR(10) REFERENCES Gym(GymID) NOT NULL,
 MembershipTypeID VARCHAR(10) REFERENCES MembershipType(MembershipTypeID) NOT NULL,
 StartDate DATE DEFAULT GETDATE(),
 ExpireDate DATE NOT NULL,
 MemberID VARCHAR(10) NOT NULL REFERENCES Member(MemberID)
);
CREATE TABLE MemberCoachCurriculum
(FlowID VARCHAR(10) PRIMARY KEY REFERENCES Flow(FlowID),
 MemberID VARCHAR(10) REFERENCES Member(MemberID),
 CurriculumID VARCHAR(10) REFERENCES Curriculum(CurriculumID),
 StaffID VARCHAR(10) REFERENCES Staff(StaffID)
);
```

Views

```
CREATE VIEW MembershipExpireDate AS
SELECT m.MemberID, m.FirstName,
m.LastName, MAX(ms.ExpireDate) AS
ExpireDate
FROM MEMBER m
INNER JOIN Membership ms
ON m.MemberID = ms.MemberID
GROUP BY m.MemberID, m.FirstName,
m.LastName;

SELECT * FROM MembershipExpireDate;
```

This view was created to show the validation of customers' membership.



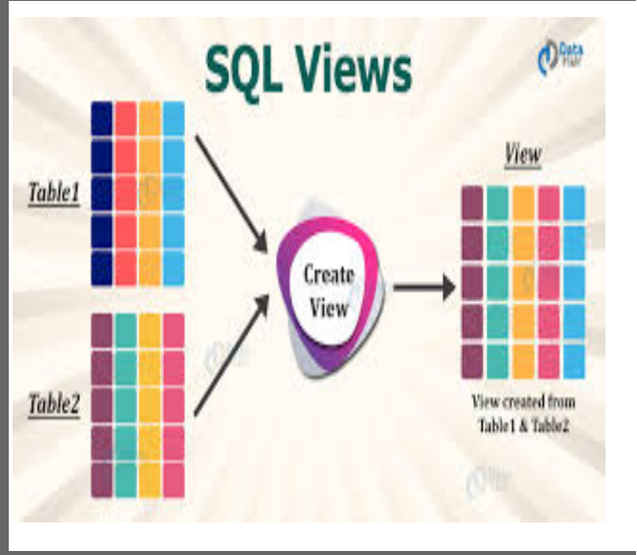
	MemberID	FirstName	LastName	ExpireDate
1	01	James	Williams	2007-05-10
2	02	Henry	Miller	2006-09-21
3	03	Alexander	Moore	2006-09-27
4	04	Charlotte	Lee	2007-04-21
5	05	Elizabeth	White	2007-07-21
6	06	Sofia	Robinson	2008-04-12
7	07	Sherry	Green	2009-04-21

Views

```
CREATE VIEW CurriculumSize AS
SELECT mcc.CurriculumID,
c.CurriculumName, mcc.StaffID,
COUNT(mcc.MemberID) AS NumOfMembers
FROM MemberCoachCurriculum mcc
INNER JOIN Curriculum c
ON mcc.CurriculumID = c.CurriculumID
GROUP BY mcc.CurriculumID,
c.CurriculumName, mcc.StaffID;

SELECT * FROM CurriculumSize;
```

This view was created to show the size of members in each curriculum.



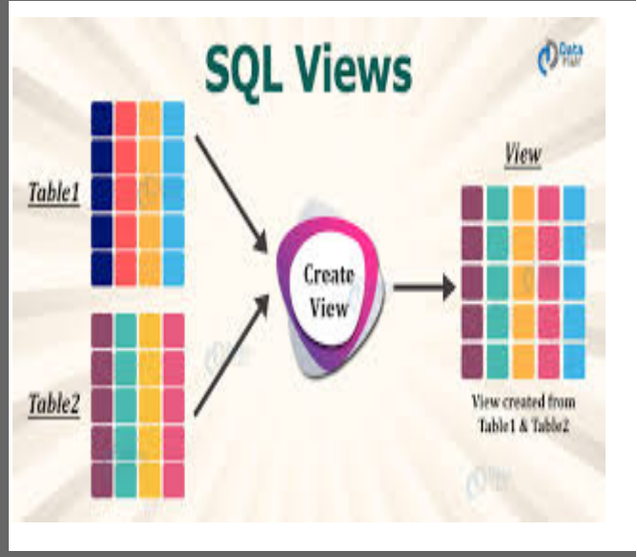
	CurriculumID	CurriculumName	NumOfMembers
1	A01	Total Body Blast	4
2	A02	Cardio Strength	3
3	A03	Triple Metcon	6
4	A04	sculpt	3
5	B01	Fat Burn Strength	3
6	B02	Ultimate	11
7	C02	Yoga sculpt	2

Views

```
CREATE VIEW CoachEvaluation AS
SELECT s.StaffID, s.GymID,
s.FirstName, s.LastName, s.Gender,
s.Age, AVG(aes.Score) AS Score
FROM Staff s
INNER JOIN AnonymousEvaluationSystem
aes
ON s.StaffID = aes.StaffID
GROUP BY s.StaffID, s.GymID,
s.FirstName, s.LastName, s.Gender,
s.Age;

SELECT * FROM CoachEvaluation;
```

This view was created to show the overall evaluation of staffs.



	StaffID	GymID	FirstName	LastName	Gender	Age	Score
1	002	01	Leonia	Hill	F	21	4.75
2	008	02	Tammy	Bailey	F	23	3.50
3	005	01	Eugene	Phillips	M	25	4.67
4	018	03	Leslie	Sealey	M	26	4.67
5	014	02	Scott	Grant	M	28	4.00
6	021	03	James	Carter	M	29	4.00
7	004	01	Theresa	Chavez	F	30	4.50

Functions

```
CREATE FUNCTION
CheckWorkLoad(@memID INT)
RETURNS INT
AS
BEGIN
DECLARE @count INT=0
SELECT @count=COUNT(MemberID)
FROM MemberCoachCurriculum
 WHERE @memID=MemberID
RETURN @count
END
```

Check if members have already registered for three lessons, they can not register any more lessons(unable to insert).

```
CREATE FUNCTION
CheckUsageTime(@equipID INT)
RETURNS INT
AS
BEGIN
RETURN (
SELECT DATEDIFF(YEAR,
f.FlowDate, GETDATE())
FROM Equipment e
JOIN Flow f
ON e.FlowID=f.FlowID
   WHERE
e.EquipmentID=@equipID
)
END;
```

Check function for retiring old equipments(unable to insert) if they are bought 5 years ago

```
CREATE FUNCTION
CheckMembershipExpired(@Member
Id varchar(10))
RETURNS SMALLINT
AS
BEGIN
DECLARE @tmp SMALLINT
SET @tmp =
CASE WHEN GETDATE() > (SELECT
MAX(ExpireDate) FROM
Membership WHERE MemberID =
          (SELECT MemberID
FROM Member WHERE MemberID =
@MemberId))
 THEN -1
 ELSE 0
END
RETURN @tmp
END
```

Check function for checking the expired membership(if expired, he/she cannot add any course anymore)

Constraints

```
GO
CREATE FUNCTION CheckUsageTime(@equipID
INT)
RETURNS INT
AS
BEGIN
RETURN (
SELECT DATEDIFF(YEAR, f.FlowDate,
GETDATE())
FROM Equipment e
JOIN Flow f
ON e.FlowID=f.FlowID
   WHERE e.EquipmentID=@equipID
)
END;
Go

ALTER TABLE Equipment WITH NOCHECK
ADD CONSTRAINT ObsoleteEquipment
CHECK
(dbo.CheckUsageTime(EquipmentID)<6);
```

We added this constraint to ensure the obsolete equipment is eliminated.


```
INSERT INTO Flow VALUES('2000', '2000-01-03')
INSERT INTO Equipment VALUES('100', '2000', '01',
'Treadmill', 300)
-- Fail to insert because of the excessive usage
time;


(1 row affected)
Msg 547, Level 16, State 0, Line 631
The INSERT statement conflicted with the CHECK constraint "ObsoleteEquipment".
The statement has been terminated.

Completion time: 2021-04-13T17:02:58.3603750-04:00
```

Constraints

```
GO
CREATE FUNCTION
CheckMembershipExpired(@MemberId
varchar(10))
RETURNS SMALLINT
AS
BEGIN
DECLARE @tmp SMALLINT
SET @tmp =
CASE WHEN GETDATE() > (SELECT
MAX(ExpireDate) FROM Membership WHERE
MemberID =
          (SELECT MemberID FROM Member WHERE
MemberID = @MemberId))
 THEN -1
 ELSE 0
END
RETURN @tmp
END
GO

ALTER TABLE MemberCoachCurriculum WITH
NOCHECK
ADD CONSTRAINT MembershipExpired CHECK
(dbo.CheckMembershipExpired(MemberID)=0);
```

We added this constraint to prevent the usage of expired membership cards


```
INSERT INTO MemberCoachCurriculum(FlowID,
MemberID, CurriculumID, StaffID) VALUES
('091', '01', 'A03', '003')
-- Fail to insert course because of the
expiration
```

```
The INSERT statement conflicted with the CHECK constraint "MembershipExpired".
The statement has been terminated.

Completion time: 2021-04-13T17:05:26.3416866-04:00
```

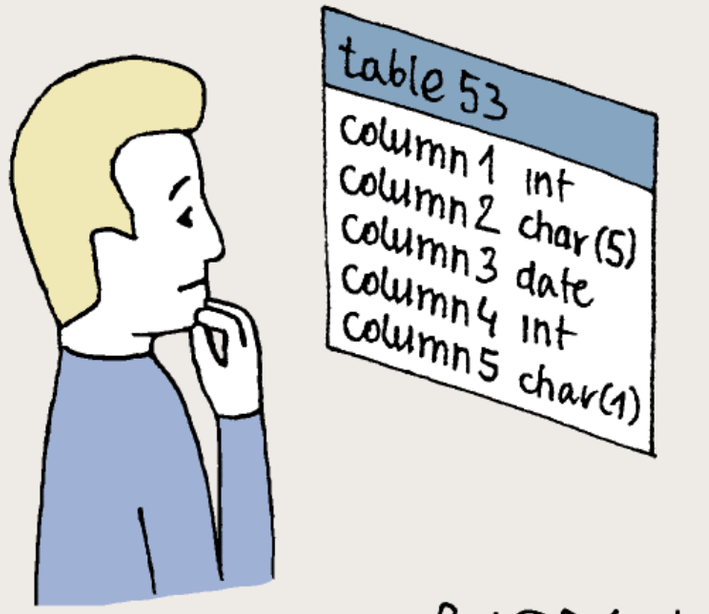
Computed Columns

```
Go
CREATE FUNCTION BMRCal(@PCID VARCHAR(10))
RETURNS INT
AS
BEGIN
    DECLARE @BMR FLOAT
      SELECT @BMR = 88.362 + (13.397 *
Weight_KG) + (4.799 * Height_CM) - (5.677 *
Age)
  FROM MemberPhysicalCondition
      WHERE PhysicalConditionID =@PCID AND
Gender = 'M'
  SELECT @BMR = 447.593 + (9.247 * Weight_KG) +
(3.098 * Height_CM) - (4.330 * Age)
  FROM MemberPhysicalCondition
  WHERE PhysicalConditionID =@PCID AND Gender =
'F'
    RETURN CAST(@BMR AS INT)
END;
Go

ALTER TABLE MemberPhysicalCondition
ADD BasalMetabolicRate AS
(dbo.BMRCal(PhysicalConditionID));

SELECT * FROM MemberPhysicalCondition;
```

We added this computed column to use data from other columns in order to calculate the basal metabolic rate of each member automatically.



Computed Columns

```
GO
CREATE FUNCTION TotalConsumptionOnCourse
(@memberid INT)
RETURNS INT
AS
BEGIN
DECLARE @tuition INT=0
SELECT @tuition=SUM(c.TuitionPerClass)
FROM MemberCoachCurriculum mcc
JOIN Curriculum c
ON mcc.CurriculumID=c.CurriculumID
WHERE mcc.MemberID=@memberid
GROUP BY mcc.MemberID
RETURN @tuition
END;
GO

ALTER TABLE Member ADD TotalConsumption AS
(dbo.TotalConsumptionOnCourse(MemberID));

SELECT * FROM Member;
```

We added this computed column to use data from other columns in order to calculate the total consumption of the each user on courses in the gym automatically.

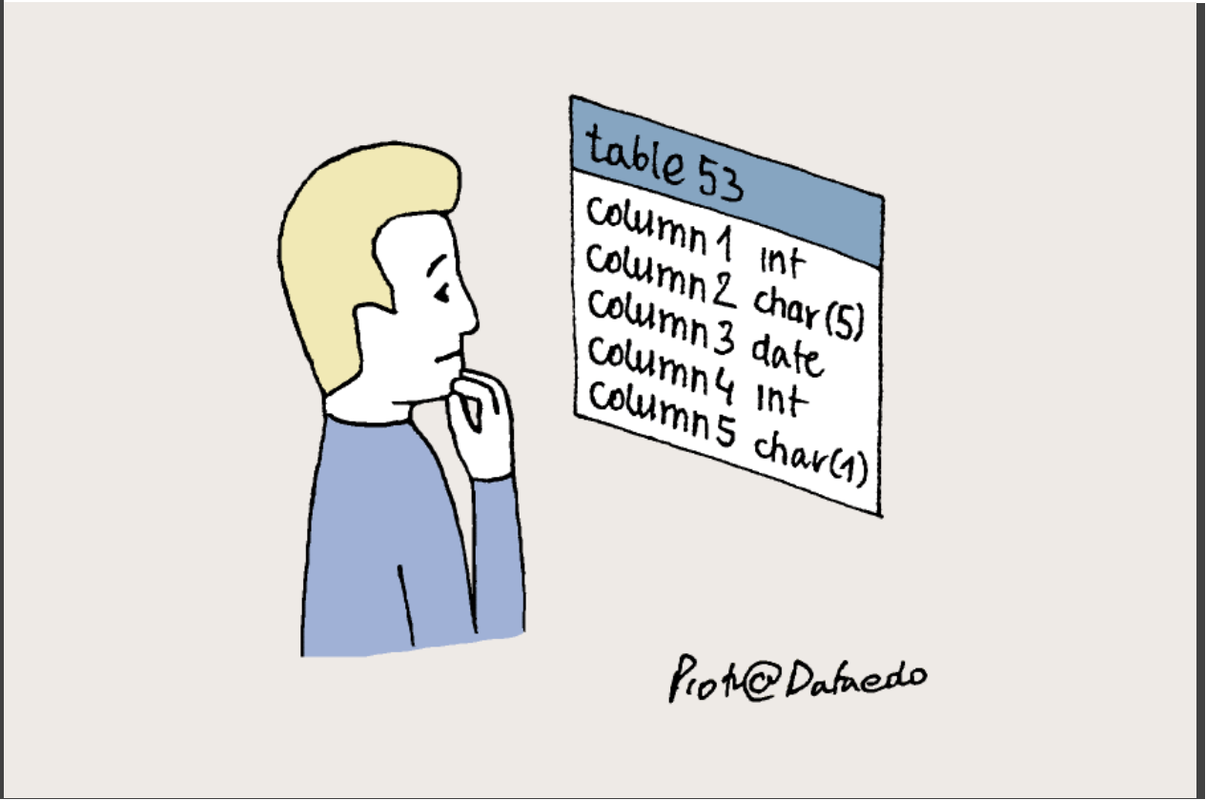


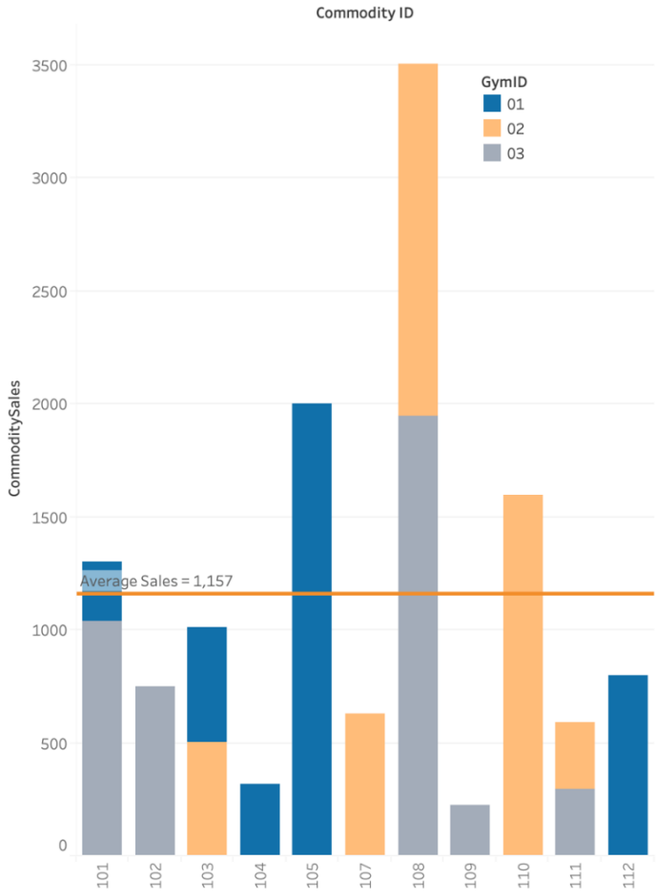
Tableau Report Visualizations&Report

Gym

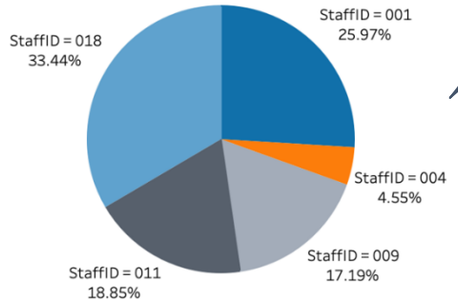
Commodity | Curriculum | Others

Identifying the sales in each commodity of GYM

Sales for Each Commodity



Commodity Sales for Each Staff



Identifying the commodity sales ratio of each staff

Tableau Report Visualizations &Report

Identifying the total cost of curriculum for each member in 3 gyms

Show the average cost of curriculum in 3 gyms

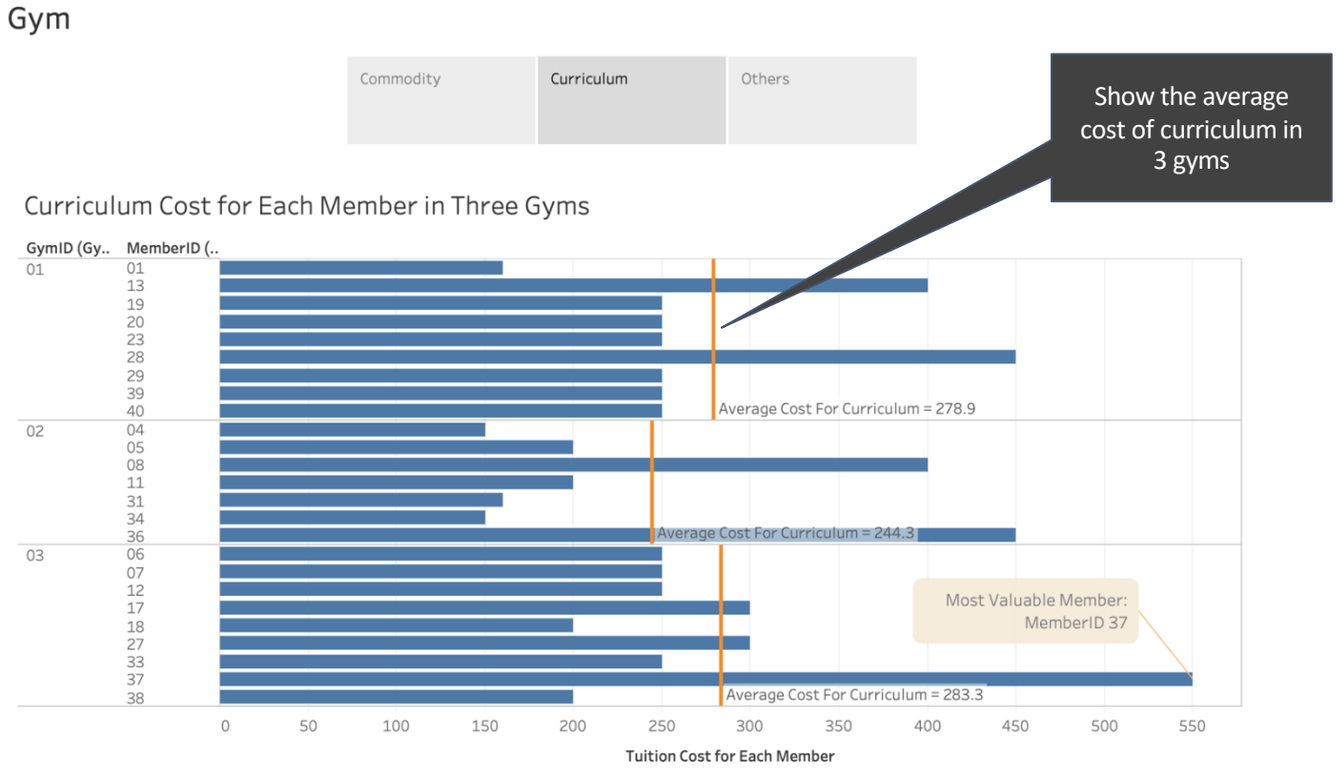


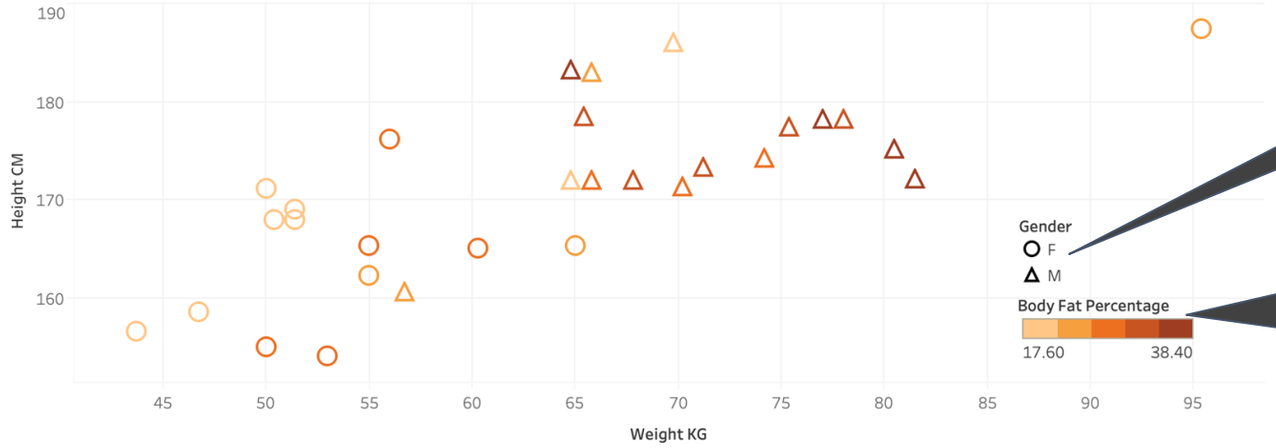
Tableau Report Visualizations &Report

Gym

Commodity | Curriculum | Others

Showing the physical condition of each member

Member Physical Condition

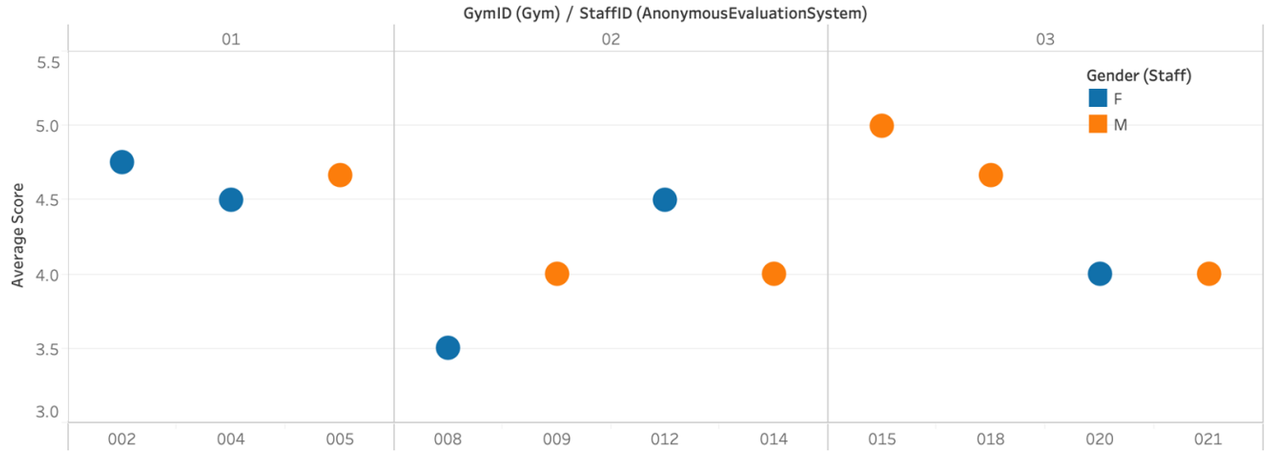


Different shapes stand for different gender

Different colors stand for different conditions of body fat

Evaluation for Staff

Generating the overall evaluation fot each staff in 3 gyms



Any Questions ?


Thank you!