Database Team Project Report

: Design database 'Korail Train booking system'

Team 6

소프트웨어학과 3학년

2014311011 김유성

2014313303 홍태하

2014314253 심건영

1. DATABASE SCHEMA

We designed train ticket in the sight of KORAIL websites. Therefore we thought about what and how to make schema.

USERINFO table saves KORAIL user's information. There are columns named userid, username, age, ismale, phonenumber, email, money, loginlog, signuplog. 'USERID' is for primary key, 'USERNAME' and 'AGE', 'ISMALE', 'LOGINLOG', and 'SIGNUPLOG' are for producing significant statistics. 'PHONENUMBER' and 'EMAIL' is for contacting users, and 'MONEY' for payment.

USERINFO								
USERID	USERNAME	AGE	ISMALE	PHONE NUMBER	EMAIL	MONEY	LOGIN LOG	SIGNUP LOG
VARCHAR2	VARCHAR2	NUMBER	NUMBER	VARCHAR2	VARCHAR2	NUMBER	VARCHAR2	VARCHAR2

Picture 01. Schema of USERINFO table

LINE table saves schedules of KORAIL train. There are columns named 'LINEID', 'STARTID', 'DESTINATIONID', 'STARTDATE', 'STARTTIME', and 'TRAINID'. 'LINEID' is for primary key, 'STARTID' and 'DESTINATIONID' referencing 'DESTINATIONID' of the **DESTINATION** table are for checking start oint and destination point. 'STARTDATE' and 'STARTTIME' are for checking train schedule, and 'TRAINID' referencing 'TRAINID' of the **TRAIN** table is for checking the train user will take on.

LINE					
LINEID	STARTID	DESTINATION ID	STARTDATE	STARTTIME	TRAINID
VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2

Picture 02. Schema of LINE table

RESERVATION table shows information of reservation. There are 'RESERVATIONID', 'USERID', 'LINEID', and 'SEATID'. 'RESERVATIONID' classifies each reservations, 'USERID' referencing 'USERID' of the **USERINFO** table is for information of user who reserve, 'LINEID' referencing 'LINEID' of the **LINE** table is for information of line which is reserved, and 'SEATID' referencing 'SEATID' of the **SEAT** table is for information of seat that means which seat user reserve.

RESERVATION			
RESERVATIONID	USERID	LINEID	SEATID
VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2

Picture 03. Schema of RESERVATION table

1. DATABASE SCHEMA

SEAT table checks available seats by each line. There are 'SEATID', 'LINEID', 'BLOCK', 'SEATNUM', 'IS_FULL', and 'SEATTYPEID'. 'SEATID' is for primary key, and 'LINEID' referencing 'LINEID' of the **LINE** table is for referencing what this seat is for. 'BLOCK' and 'SEATNUM' are the information of seat position. 'IS_FULL' tells the seat is now available or not, and 'SEATTYPEID' referencing 'SEATTYPEID' of the **SEATTYPE** table tells seat's rating.

SEATTYPE table saves rating of seats. 'SEATTYPEID' is for primary key, and 'SEATTYPE' stores the rating of seats.

SEATPRICE table saves prices of seats by seat types. 'SEATPRICEID' is for primary key, and 'SEATTYPEID' referencing 'SEATTYPEID' of the **SEATTYPE** table distinguishes each seats. 'PRICE' stores the price by rating of seats.

SEAT					
SEATID	LINEID	BLOCK	SEATNUM	IS_FULL	SEATTYPE ID
VARCHAR2	VARCHAR2	NUMBER	NUMBER	NUMBER	VARCHAR2

SEATTYPE	
SEATTYPEID	SEATTYPE
VARCHAR2	VARCHAR2

SEATPRICE		
SEATPRICEID	SEATTYPEID	PRICE
VARCHAR2	VARCHAR2	NUMBER

Picture 04. Schemas of SEAT table, SEATTYPE table, and SEATPRICE table

TRAIN table gets information of train. There are 'TRAINID', 'TRAINNAME', and 'TRAINTYPEID'. 'TRAINID' is primary key for separate each tuple, 'TRAINNAME' is name of train, and 'TRAINTYPEID' referencing 'TRAINTYPEID' of the **TRAINTYPE** table is for get information of train type.

TRAINTYPE table is for get information of train type. There are 'TRAINTYPEID', and 'TRAINTYPE'. 'TRAINTYPEID' is primary key for separate each tuple, 'TRAINTYPE' has two tuples that 'mugunghwa', and 'saemaul'.

TRAINPRICE table saves prices of seats by seat types. 'TRAINPRICEID' is for primary key, and 'TRAINTYPEID' referencing 'TRAINTYPEID' of the **TRAINTYPE** table is for distinguishing train type. 'PRICE' stores the price on the rating of seats.

TRAIN		
TRAINID	TRAINNAME	TRAINTYPEID
VARCHAR2	VARCHAR2	VARCHAR2

TRAINTYPE	
TRAINTYPEID	TRAINTYPE
VARCHAR2	VARCHAR2

TRAINPRICE		
TRAINPRICEID	TRAINTYPEID	PRICE
VARCHAR2	VARCHAR2	NUMBER

Picture 05. Schemas of TRAIN table, TRAINTYPE table, and TRAINPRICE table

1. DATABASE SCHEMA

DESTINATION table is for get information of destination. There are ‘DESTINATIONID’, and ‘DESTINATIONNAME’. ‘DESTINATIONID’ is primary key for separate each tuple, and ‘DESTINATIONNAME’ is information of name of each destination.

DESTINATION	
DESTINATIONID	DESTINATIONNAME
VARCHAR2	VARCHAR2

Picture 06. Schema of DESTINATION TABLE

DISTANCE table saves the distance from point to point. ‘DISTANCEID’ is for primary key. ‘DESTINATIONID1’ means the start point and ‘DESTINATIONID2’ means the destination point. Each ‘DESTINATIONID’ columns reference ‘DESTINATIONID’ of the **DESTINATION** table. ‘DISTANCE’ column stores distance values.

DISTANCE			
DISTANCEID	DESTINATION ID1	DESTINATION ID2	DISTANCE
VARCHAR2	VARCHAR2	NUMBER	NUMBER

Picture 07. Schema of DISTANCE table

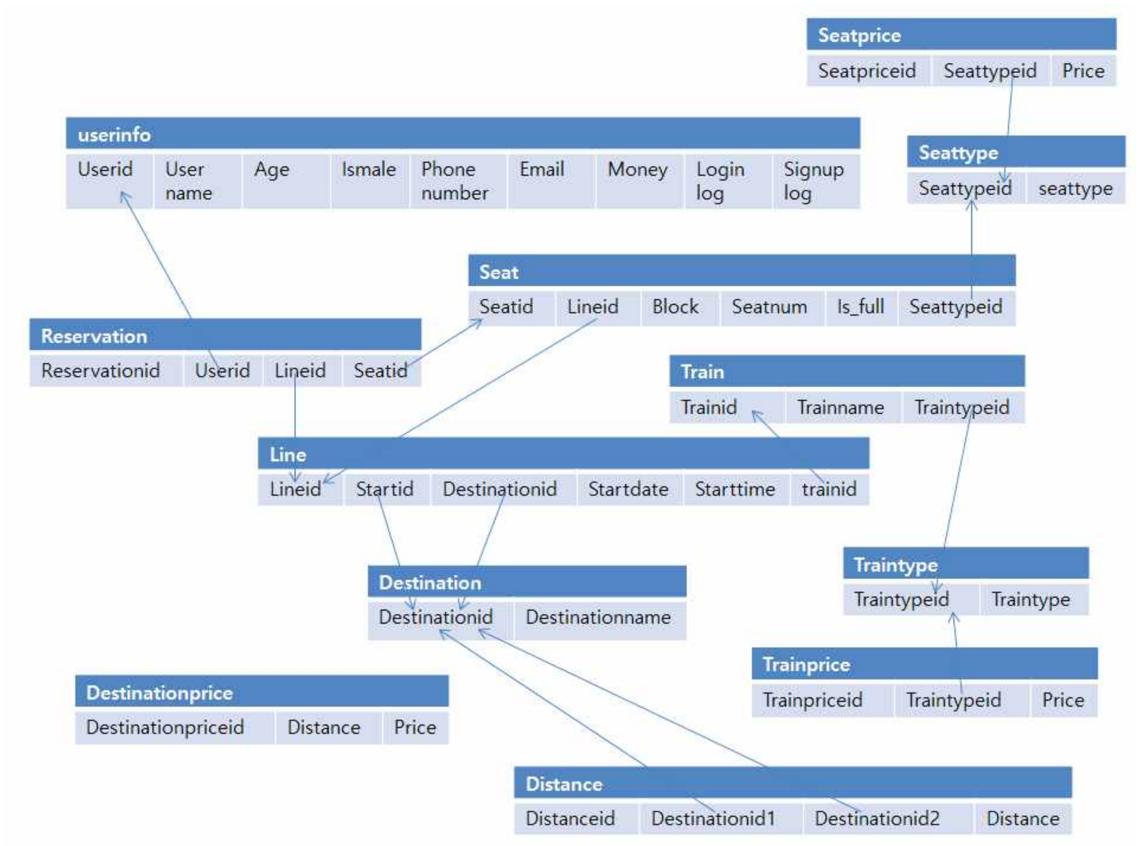
DESTINATIONPRICE table saves the prices on each distances. ‘DESTINATIONPRICEID’ is for primary key. ‘DISTANCE’ stores the distances and ‘PRICE’ stores the prices.

DESTINATIONPRICE		
DESTINATIONP RICEID	DISTANCE	PRICE
VARCHAR2	VARCHAR2	NUMBER

Picture 08. Schema of DESTINATIONPRICE table

2. DESCRIBE RELATIONSHIP BETWEEN TABLES

RESERVATION table refer **USERINFO**, **LINE**, and **SEAT** table for get information of user, seat, and line. **RESERVATION** can know who reserve, which seat is reserved, and which line is reserved. **SEAT** table refer **SEATTYPE** table for get information of type of seat. **TRAIN** table refer to **TRAINTYPE** table for get information of train type. It means that what is this train. **TRAINPRICE** table also refer to **TRAINTYPE** table for get information of train type. By this information **TRAINPRICE** table get price of each train type. **LINE** table refer to **DESTINATION** table and **TRAIN** table. In **LINE** table, there are name of start city, and destination city, so **DESTINATION** table send the information of name of cities. **TRAIN** table send the information of train type. **LINE** table need to know which train run on this line. **DISTANCE** table refer to **DESTINATION** table. **DISTANCE** table need to know two cities' id. **DISTANCE** table show distance of these cities.



Picture 09. Relations of Entities

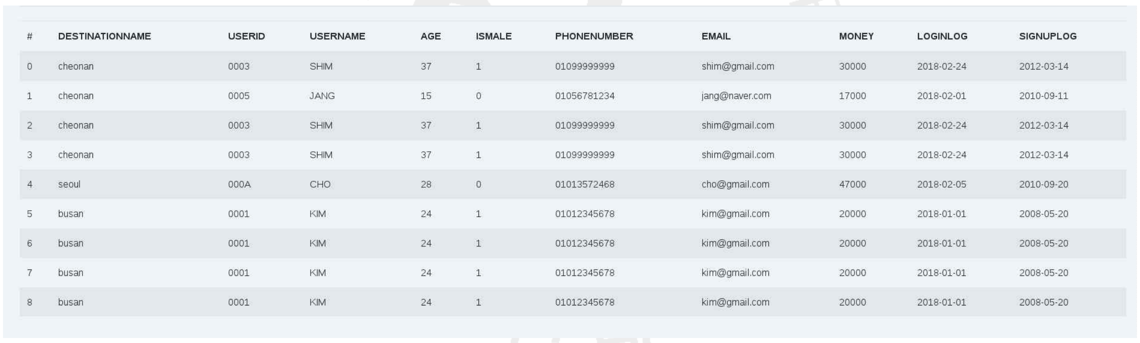
3. PROBLEM AND SQL STATEMENT

JOIN>

```
SELECT D.DESTINATIONNAME, U.*
FROM USERINFO U, LINE L, RESERVATION R, DESTINATION D
WHERE U.USERID=R.USERID
        AND L.LINEID=R.LINEID
        AND L.DESTINATIONID = D.DESTINATIONID
```

This is query for get information of users who go to specific destination. This query show the name of destination and all of **USERINFO** table. We use join to four tables, **USERINFO**, **LINE**, **RESERVATION**, and **DESTINATION**, and we can get who reserve the train, which line user reserve, and where is line's destination. 'USERID' is primary key of **USERINFO** table, and also foreign key of **RESERVATION** table. 'LINEID' is primary key of **LINE** table, and also foreign key of **RESERVATION** table. DESTINATIONID is primary key of **DESTINATION** table and also foreign key of **LINE** table. These tables are connected these attributes so we can use these tables by join.

Result



#	DESTINATIONNAME	USERID	USERNAME	AGE	ISMALE	PHONENUMBER	EMAIL	MONEY	LOGINLOG	SIGNUPLOG
0	cheonan	0003	SHIM	37	1	01099999999	shim@gmail.com	30000	2018-02-24	2012-03-14
1	cheonan	0005	JANG	15	0	01056781234	jang@naver.com	17000	2018-02-01	2010-09-11
2	cheonan	0003	SHIM	37	1	01099999999	shim@gmail.com	30000	2018-02-24	2012-03-14
3	cheonan	0003	SHIM	37	1	01099999999	shim@gmail.com	30000	2018-02-24	2012-03-14
4	seoul	000A	CHO	28	0	01013572468	cho@gmail.com	47000	2018-02-05	2010-09-20
5	busan	0001	KIM	24	1	01012345678	kim@gmail.com	20000	2018-01-01	2008-05-20
6	busan	0001	KIM	24	1	01012345678	kim@gmail.com	20000	2018-01-01	2008-05-20
7	busan	0001	KIM	24	1	01012345678	kim@gmail.com	20000	2018-01-01	2008-05-20
8	busan	0001	KIM	24	1	01012345678	kim@gmail.com	20000	2018-01-01	2008-05-20

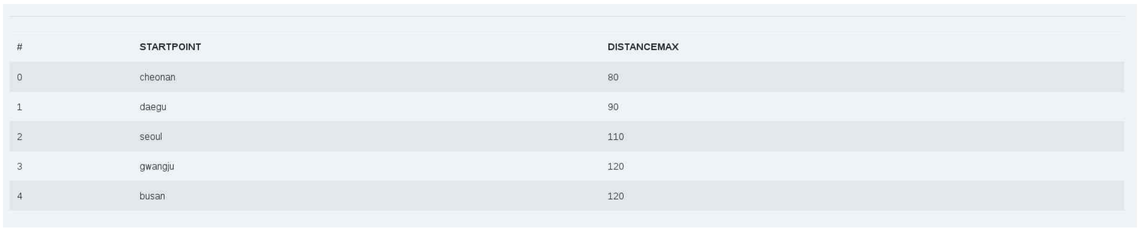
SORTING WITH GROUP BY FUNCTION>

```
SELECT D.DESTINATIONNAME AS STARTPOINT, MAX(DS.DISTANCE) AS DISTANCEMAX
FROM DESTINATION D, DISTANCE DS
WHERE D.DESTINATIONID = DS.DESTINATIONID1
GROUP BY D.DESTINATIONNAME
ORDER BY DISTANCEMAX
```

This query calculates and shows the most far destination's distance from the specific point. We used tables named **DESTINATION** and **DISTANCE**, group by function and sorted it by distances. We can use this query to calculate the maximum distances from point to point and make efficiency running schedules.

3. PROBLEM AND SQL STATEMENT

Result



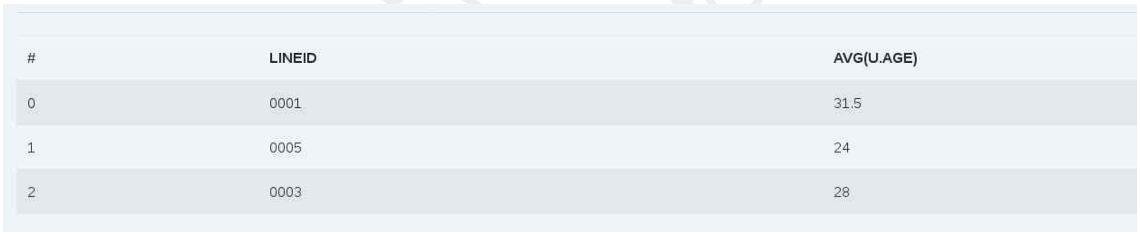
#	STARTPOINT	DISTANCEMAX
0	cheonan	80
1	daegu	90
2	seoul	110
3	gwangju	120
4	busan	120

AGGREGATE FUNCTION>

```
SELECT L.LINEID, AVG(U.AGE)
FROM LINE L, USERINFO U, RESERVATION R
WHERE L.LINEID=R.LINEID AND U.USERID=R.USERID
GROUP BY L.LINEID
```

This is query for get user's average age in each line. This query shows id of line and average age of users. We use aggregate function 'AVG', and join three tables, **LINE**, **USERINFO**, and **RESERVATION**. We want to get information of just user who use each line, so where clause means that. Also, we want to show average age and id of line too, so we use 'group by'. People who watch this information can know that which line is older people's favorite or younger people's favorite. We can use this information for advertising something for older people, or get vacation place that youngest mostly visit.

Result



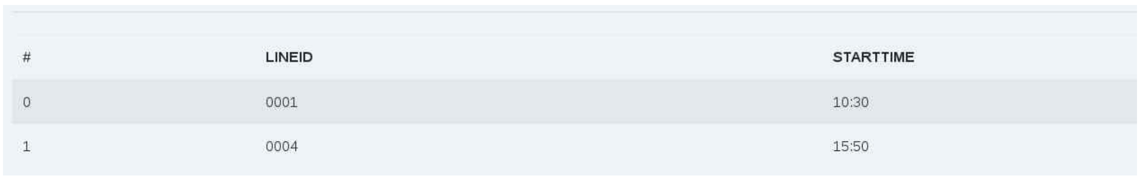
#	LINEID	AVG(U.AGE)
0	0001	31.5
1	0005	24
2	0003	28

SUB-QUERY>

```
SELECT L.LINEID, L.STARTTIME
FROM LINE L
WHERE L.TRAINID
          IN (SELECT T.TRAINID FROM TRAIN T, TRAINTYPE P
          WHERE T.TRAINNAME=P.TRAINTYPE)
```

This is query for get line the train 'saemaeul' go. This query shows id of line and its start time. We use sub query for get that information. In sub query, we can get train's id whose name is same to **TRAINTYPE** table's train type. In our dummy data, there are two 'mugunghwa' and one 'saemaul', so in **TRAIN** table, 'mugunghwa's name is 'mugunghwa1', and 'mugunghwa2' so just 'saemaul' is commonly in **TRAIN** table's 'TABLENAME' and **TRAINTYPE** table's 'TABLETYPE'. So in sub query, we can get 'saemaul's 'TRAINID's. If line's 'TRAINID' is in that ids, that line is run by 'saemaul'. Commonly, 'saemaul' is better than 'mugunghwa', people who want to travel by better train can use this information.

Result



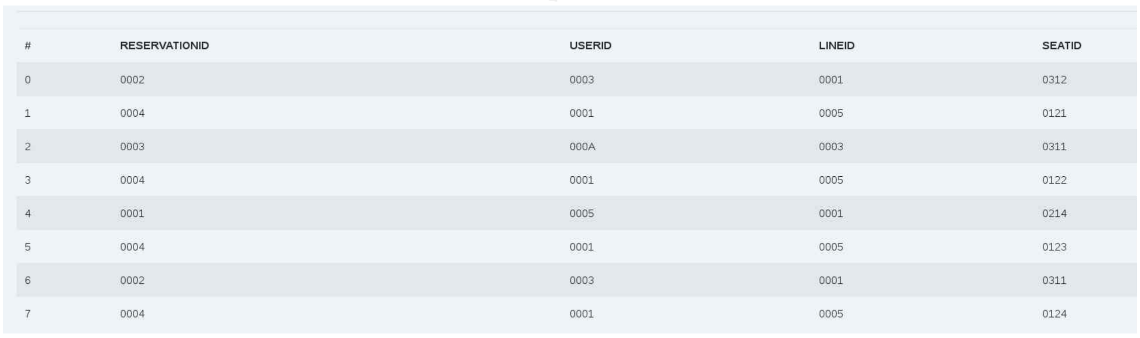
#	LINEID	STARTTIME
0	0001	10:30
1	0004	15:50

DUPLICATE>

```
SELECT DISTINCT * FROM RESERVATION
```

This is query that remove duplicate tuples In our dummy data, 'RESERVATION' table doesn't have primary key so there can be some duplicate data. 'DISTINCT' means there cannot be same tuples. In reservation, there are information of user, line, and seat, and same time, same line, and same train's seat can be just reserved one user so duplicate tuple is meaningless. No one want to know unnecessary data so we remove duplicate.

Result



#	RESERVATIONID	USERID	LINEID	SEATID
0	0002	0003	0001	0312
1	0004	0001	0005	0121
2	0003	000A	0003	0311
3	0004	0001	0005	0122
4	0001	0005	0001	0214
5	0004	0001	0005	0123
6	0002	0003	0001	0311
7	0004	0001	0005	0124

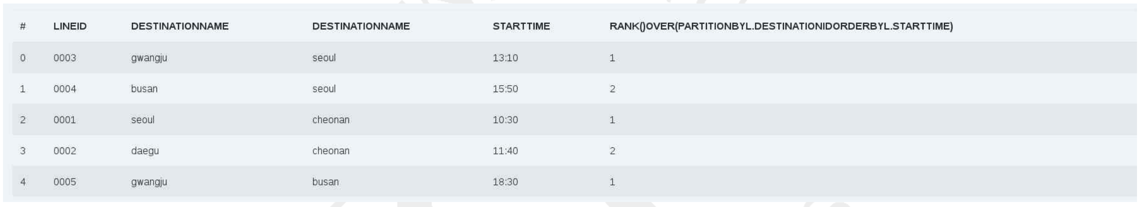
3. PROBLEM AND SQL STATEMENT

OWN PROBLEM 1>

```
SELECT L.LINEID, D2.DESTINATIONNAME, D1.DESTINATIONNAME, L.STARTTIME,
RANK() OVER(PARTITION BY L.DESTINATIONID ORDER BY L.STARTTIME)
FROM LINE L, DESTINATION D1, DESTINATION D2
WHERE L.DESTINATIONID = D1.DESTINATIONID AND L.STARTID=D2.DESTINATIONID
ORDER BY L.DESTINATIONID
```

This is query for get start time for each destination. This query show id of line, name of start city, name of destination, start time, and rank of start time. We want to show information of start time for each destination. So in rank function, there is partition that is for separate tuples by destination. By this information, people who want to go specific city can know which one is the fastest in their start city.

Result



#	LINEID	DESTINATIONNAME	DESTINATIONNAME	STARTTIME	RANK()OVER(PARTITIONBYL.DESTINATIONIDORDERBYL.STARTTIME)
0	0003	gwangju	seoul	13:10	1
1	0004	busan	seoul	15:50	2
2	0001	seoul	cheonan	10:30	1
3	0002	daegu	cheonan	11:40	2
4	0005	gwangju	busan	18:30	1

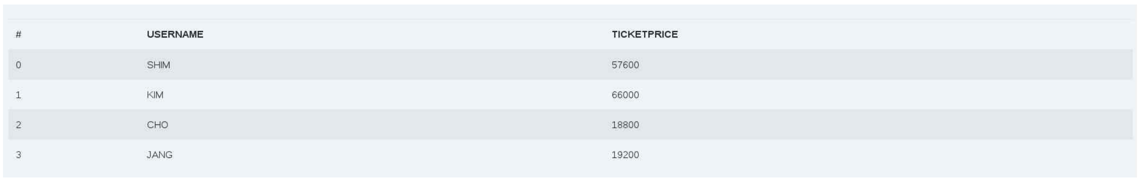
OWN PROBLEM 2>

```
SELECT U.USERNAME AS USERNAME, SUM(TP.PRICE + DP.PRICE + SP.PRICE)/2
          AS TICKETPRICE
FROM USERINFO U, RESERVATION R, TRAIN T, TRAINTYPE TT, DESTINATION D1,
DESTINATION D2, DISTANCE DS, SEAT S, DESTINATIONPRICE DP, TRAINPRICE TP,
SEATPRICE SP, LINE L
WHERE R.LINEID = L.LINEID
          AND L.STARTID = D1.DESTINATIONID
          AND L.DESTINATIONID = D2.DESTINATIONID
          AND D1.DESTINATIONID = DS.DESTINATIONID1
          AND D2.DESTINATIONID = DS.DESTINATIONID2
          AND DS.DISTANCE = DP.DISTANCE
          AND L.TRAINID = T.TRAINID
          AND T.TRAINTYPEID = TP.TRAINTYPEID
          AND R.SEATID = S.SEATID
          AND S.SEATTYPEID = SP.SEATTYPEID
          AND U.USERID = R.USERID
GROUP BY U.USERNAME
```

3. PROBLEM AND SQL STATEMENT

This query calculates and shows how much money each registered user in korail spent. It shows each user's name and sum of money they used. We used some joins with tables named **USERINFO**, **RESERVATION**, **TRAIN**, **TRAINTYPE**, **DESTINATION**, **DISTANCE**, **SEAT**, **DESTINATIONPRICE**, **TRAINPRICE**, **SEATPRICE** and **LINE**, aggregation 'SUM' and numerical expressions. We can use this query for giving benefits to users as how much money users spent.

Result



#	USERNAME	TICKETPRICE
0	SHIM	57600
1	KIM	66000
2	CHO	18800
3	JANG	19200