Salary Manager - Project

Java, MySQL, JDBC – A Console Driven Project

Complete Project Link: https://drive.google.com/open?id=1FHaO2i3R_CozuqmLbvb9Xs5sWoyDqYcE

Project Working Demo Video: https://youtu.be/JTzFz7wpwvk

Things I did

- Designed everything from scratch. Starting from Modularizing the code blocks using OOPS concepts to designing the database (Please view my github repo)
- Github repo link: https://github.com/vivekVells/Salary-Manager-Java-MySql-Project/
- Please refer the project sourcecode header below

Objective

- To track and manage the income and expense transaction of a person

Languages Used

- Java
- MySql (via JAVA JDBC Connectivity)

Features

- Create an User Account
- Enter Income or Expense Information
- View Income or Expense or Net Balance Summary Information

Project SourceCode:

- GitHub link: https://github.com/vivekVells/Salary-Manager-Java-MySql-Project/
- Complete Project Drive : https://drive.google.com/open?id=1FHaO2i3R_CozuqmLbvb9Xs5sWoyDqYcE
- Front End: https://drive.google.com/open?id=1qHulfgTIDBBVd1oDGj2quJUzWANDDrtQ
- Back End: https://drive.google.com/open?id=1ehqKz-c4OUcgvEBmRTBchYlV-VQd4BYX

Concepts Learnt & Practiced

- Efficience & Importance of DataBase Designing
    - DML, DDL, Triggers and Variables
    - Normalization & Bridge/Junction
    - How to avoid data redundancy and mantain data integrity
    - Entity Relationship Diagram
- OOPS concepts
    - Modularity of the code blocks which enabled to understand the value of code reusability
- JDBC stuffs
    - Connection Establishing, Statement Creation, ResultSet retrival
- How to export and execute a runnable Jar file
- How to clear previous output in the windows console
- How to send email using javax.jar file

Future Code

- Little tweaks
- Moving to better GUI

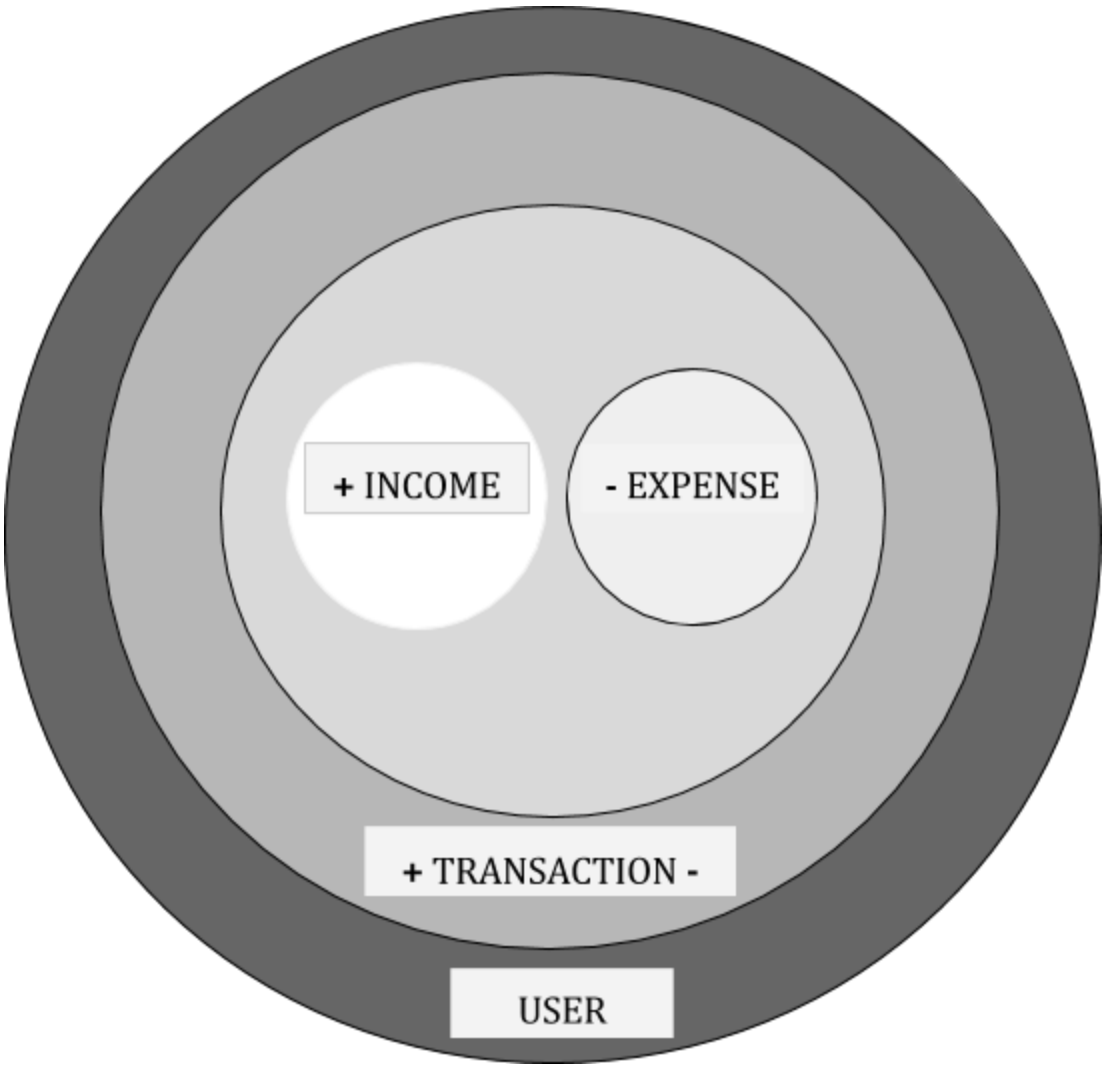
Consider the followings

- Database Design
- ER Diagram
- Data Dictionary

Salary Manager

To manage the transactions that involves in earning and spending the money. Kindly refer the diagram…

From Outer to Inner View:



Notables: Gist

- All the tables are made independent on each other and made in such a way that it wont hold any redundant data. This is achieved by the use of the junction or bridge tables and normalization concepts.
- Triggers, key constraints, if conditions and variables are also used.
- Log maintenance is given importance. All the user login information is stored separately. Similarly, all the table creation, updation and deletion are maintained.
- Tables are separated into blocks so that the purpose can be well identified. The blocks are
    1. Independent Table Block
        a. Address
    2. User Profile Table Block
        a. User
        b. User_Info
        c. User_Address
        d. User_Log
    3. Transaction Table Block
        a. Transaction
        b. Income_Transaction
        c. Expense_Transaction
    4. Earning Table Block
        a. Job
        b. Income
    5. Expense Table Block
        a. Expense

Scenario:

1) User Log In
    a) Tables involded:
        i) User
            (1) To maintain user account profiles
        ii) UserInfo
            (1) To maintain additional user information
        iii) UserAddress
            (1) To maintain different address used by user

2) Transaction Updation
    a) Tables involved
        i) Transaction
            (1) All transaction information will be available here
        ii) Income_transaction
            (1) All income transaction will be available here

iii) Expense_transaction

(1) All expense transaction will be available here

b) Earning via jobs

i) E-g: Employee working as a 'Data Scientist' (Job type 1) and then working as part time driver in Uber (Job type 2) that sums up the income

ii) Tables involved:

(1) Income

(a) Salary earned by which job - such information avail here

(2) Job

(a) Job types

c) Expense

i) Table involved:

(1) Expense

(a) Expense spent after earning info avail

d) Independent Tables

i) Tables involved:

(1) Address

(a) This table will be the hub to store all address entries

(b) In future, job addresses will also be using this one.

Concepts Review while designing the database:

Income Table (consider this for an example)

Job table

jobId	roleName	incomeAmount	earnedDate	note	**userId**
45	Desk Asst	3000.00	11/02/2017	Job @desk	3
46	Uber	2500.00	11/04/2017	job@ride	5
47	Web Design	1200.00	11/07/2017	job@design	3
48	Online Tutoring	1700.00	11/11/2017	lyft@ny	3
49	Desk Asst	1300.00	11/05/2017	dek@asst	5
50	Web Design	4100.00	11/21/2017	web@des	5

- **Observed**
  - 1NF achieved since atomic values alone present (i.e) only one attrib values present aptly.
  - 2NF achieved since all the entries available there depends upon the primary key jobId like jobRoleName, amount earned based on job, amount earned because of the job at a date, notes to note regarding job which depends upon job and finally, the userId who is mapped to jobId.
  - 3NF not achieved since, all the non-prime members are dependent on each other and data redundancy available which is not a good design. So, we split the tables accordingly now.

Income table

incomeId	incomeAmount	earnedDate	note	**userId**	**jobId**
63	3000.00	11/02/2017	Job @desk	3	45
64	2500.00	11/04/2017	job@ride	5	46
65	1200.00	11/07/2017	job@design	3	47
66	1700.00	11/11/2017	ont@ny	3	48
67	1300.00	11/05/2017	dek@asst	5	49
68	4100.00	11/21/2017	web@des	5	50

Job table

jobId	roleName
45	Desk Asst
46	Uber
47	Web Design
48	Online Tutoring

- 3NF though achieved, 3.5NF or BCNF can be used to further decompose this table to access the entries we desire faster. 3.5NF deals with the multi valued dependncies of primary keyed values. Here, jobId, userId and incomeId are in a single table. That can further be reduced to different tables. We can use the junction table and move the userId to **Transaction Table** (this is done

because same scenario ocurred for **Expense table** as only one userId column when added to transaction table avoided the chance of including it into **Income and Expense table**).

Income table

incomeId	incomeAmount	earnedDate	note	**jobId**	**userId**
63	3000.00	11/02/2017	Job @desk	45	3
64	2500.00	11/04/2017	job@ride	46	5
65	1200.00	11/07/2017	job@design	47	3
66	1700.00	11/11/2017	ont@ny	48	3
67	1300.00	11/05/2017	dek@asst	49	5
68	4100.00	11/21/2017	web@des	50	3

IncomeTransaction table (Junction or Bridge table)

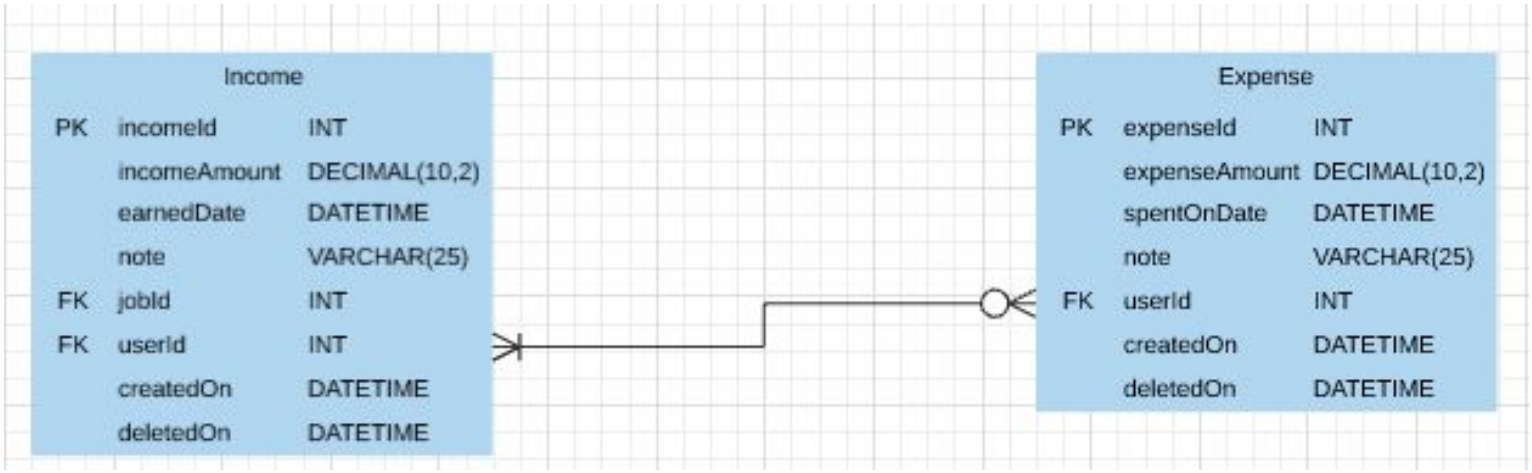
incomeTransactionId	**transactionId**	**incomeId**
52	84	63
53	85	64
54	86	65
55	87	66
56	88	67
57	89	68

Transaction table

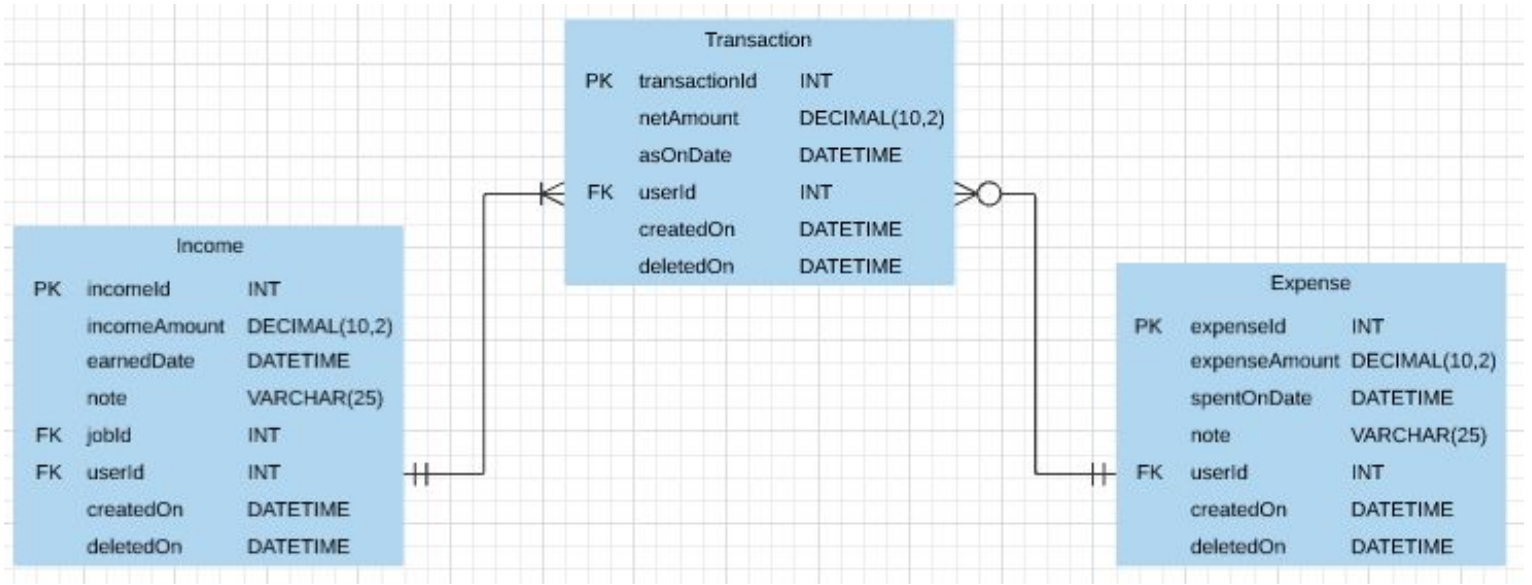
transactionId	**userId**	netAmount	asOnDate	isIncome
84	3	3000.00	11/02/2017	Y
85	5	2500.00	11/04/2017	Y
86	3	4200.00	11/07/2017	Y
87	3	5900.00	11/11/2017	Y

- Also, **Transaction table** behaves as **Bridge table** between **Job+Income** and **Expense table** since many to many relationship between job/income and expense table observed.

Before Bridge Table



After Bridge Table



Future Improvements:

- Job address addition along with location access (on click, will give from: current_location to: job address)
- Loan from others and Loan to others feature to be added
- User profile image feature
- User email password recovery feature

Salary Manager ER Diagram

Block 1

userinfo
- userInfoId INT(11)
- userId INT(11)
- firstName VARCHAR(25)
- middleName VARCHAR(25)
- lastName VARCHAR(25)
- phoneNumber BIGINT(20)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes

userlog
- userLogId INT(11)
- userId INT(11)
- lastLoggedIn DATETIME
- lastLoggedOut DATETIME
- deletedOn DATETIME
- Indexes
- Triggers

user_address
- userAddressId INT(11)
- userId INT(11)
- addressId INT(11)
- Indexes

address
- addressId INT(11)
- street VARCHAR(25)
- city VARCHAR(20)
- state CHAR(2)
- zipcode INT(11)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes

USER Block

user
- userId INT(11)
- username VARCHAR(10)
- password VARCHAR(10)
- recoveryAnswer VARCHAR(45)
- recoveryEmail VARCHAR(30)
- createdOn DATETIME
- lastUpdatedOn DATETIME
- deletedOn DATETIME
- Indexes

Block 2

TRANSACTION Block

income_transaction
- incomeTransactionId INT(11)
- incomeId INT(11)
- transactionId INT(11)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes

transaction
- transactionId INT(11)
- netAmount DECIMAL(10,2)
- asOnDate DATETIME
- userId INT(11)
- isIncome CHAR(1)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes
- Triggers

expense_transaction
- expenseTransactionId INT(11)
- expenseId INT(11)
- transactionId INT(11)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes

Block 3

INCOME Block

income
- incomeId INT(11)
- incomeAmount DECIMAL(10,2)
- earnedDate DATE
- note VARCHAR(25)
- jobRole VARCHAR(25)
- userId INT(11)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes
- Triggers

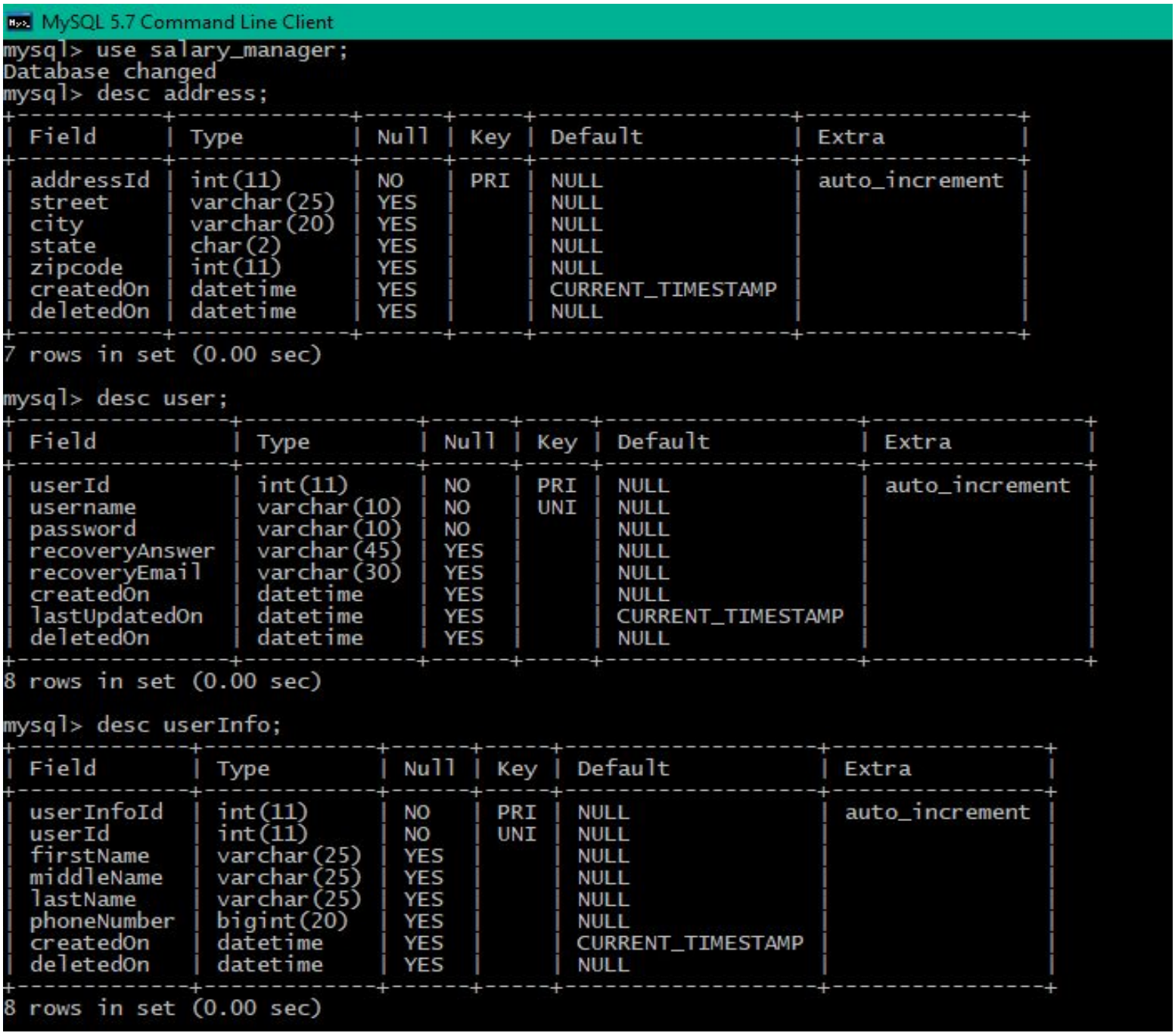
job
- jobId INT(11)
- roleName VARCHAR(25)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes

Block 4

EXPENSE Block

expense
- expenseId INT(11)
- expenseAmount DECIMAL(10,2)
- spentOnDate DATE
- note VARCHAR(25)
- userId INT(11)
- createdOn DATETIME
- deletedOn DATETIME
- Indexes
- Triggers

Salary Manager - Data Dictionary



Salary Manager - Data Dictionary

MySQL 5.7 Command Line Client

```
mysql> desc user_address;
+---------------+---------+------+-----+---------+----------------+
| Field         | Type    | Null | Key | Default | Extra          |
+---------------+---------+------+-----+---------+----------------+
| userAddressId | int(11) | NO   | PRI | NULL    | auto_increment |
| userId        | int(11) | NO   | MUL | NULL    |                |
| addressId     | int(11) | NO   | MUL | NULL    |                |
+---------------+---------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> desc userLog;
+---------------+----------+------+-----+-------------------+----------------+
| Field         | Type     | Null | Key | Default           | Extra          |
+---------------+----------+------+-----+-------------------+----------------+
| userLogId     | int(11)  | NO   | PRI | NULL              | auto_increment |
| userId        | int(11)  | NO   | MUL | NULL              |                |
| lastLoggedIn  | datetime | YES  |     | CURRENT_TIMESTAMP |                |
| lastLoggedOut | datetime | YES  |     | NULL              |                |
| deletedOn     | datetime | YES  |     | NULL              |                |
+---------------+----------+------+-----+-------------------+----------------+
5 rows in set (0.00 sec)

mysql> desc transaction;
+---------------+--------------------------+------+-----+-------------------+----------------+
| Field         | Type                     | Null | Key | Default           | Extra          |
+---------------+--------------------------+------+-----+-------------------+----------------+
| transactionId | int(11)                  | NO   | PRI | NULL              | auto_increment |
| netAmount     | decimal(10,2) unsigned   | NO   |     | 0.00              |                |
| asOnDate      | datetime                 | NO   |     | CURRENT_TIMESTAMP |                |
| userId        | int(11)                  | NO   | MUL | NULL              |                |
| isIncome      | char(1)                  | YES  |     | NULL              |                |
| createdOn     | datetime                 | YES  |     | CURRENT_TIMESTAMP |                |
| deletedOn     | datetime                 | YES  |     | NULL              |                |
+---------------+--------------------------+------+-----+-------------------+----------------+
7 rows in set (0.00 sec)

mysql> desc income_transaction;
+---------------------+----------+------+-----+-------------------+----------------+
| Field               | Type     | Null | Key | Default           | Extra          |
+---------------------+----------+------+-----+-------------------+----------------+
| incomeTransactionId | int(11)  | NO   | PRI | NULL              | auto_increment |
| incomeId            | int(11)  | NO   | MUL | NULL              |                |
| transactionId       | int(11)  | NO   | MUL | NULL              |                |
| createdOn           | datetime | YES  |     | CURRENT_TIMESTAMP |                |
| deletedOn           | datetime | YES  |     | NULL              |                |
+---------------------+----------+------+-----+-------------------+----------------+
5 rows in set (0.00 sec)
```

Salary Manager - Data Dictionary

MySQL 5.7 Command Line Client

```
mysql> desc expense_transaction;
+----------------------+----------+------+-----+-------------------+----------------+
| Field                | Type     | Null | Key | Default           | Extra          |
+----------------------+----------+------+-----+-------------------+----------------+
| expenseTransactionId | int(11)  | NO   | PRI | NULL              | auto_increment |
| expenseId            | int(11)  | NO   | MUL | NULL              |                |
| transactionId        | int(11)  | NO   | MUL | NULL              |                |
| createdOn            | datetime | YES  |     | CURRENT_TIMESTAMP |                |
| deletedOn            | datetime | YES  |     | NULL              |                |
+----------------------+----------+------+-----+-------------------+----------------+
5 rows in set (0.00 sec)

mysql> desc job;
+-----------+-------------+------+-----+-------------------+----------------+
| Field     | Type        | Null | Key | Default           | Extra          |
+-----------+-------------+------+-----+-------------------+----------------+
| jobId     | int(11)     | NO   | PRI | NULL              | auto_increment |
| roleName  | varchar(25) | NO   |     | NULL              |                |
| createdOn | datetime    | YES  |     | CURRENT_TIMESTAMP |                |
| deletedOn | datetime    | YES  |     | NULL              |                |
+-----------+-------------+------+-----+-------------------+----------------+
4 rows in set (0.00 sec)

mysql> desc income;
+--------------+---------------+------+-----+-------------------+----------------+
| Field        | Type          | Null | Key | Default           | Extra          |
+--------------+---------------+------+-----+-------------------+----------------+
| incomeId     | int(11)       | NO   | PRI | NULL              | auto_increment |
| incomeAmount | decimal(10,2) | NO   |     | NULL              |                |
| earnedDate   | datetime      | NO   |     | NULL              |                |
| note         | varchar(25)   | YES  |     | NULL              |                |
| jobId        | int(11)       | NO   | MUL | NULL              |                |
| userId       | int(11)       | NO   | MUL | NULL              |                |
| createdOn    | datetime      | YES  |     | CURRENT_TIMESTAMP |                |
| deletedOn    | datetime      | YES  |     | NULL              |                |
+--------------+---------------+------+-----+-------------------+----------------+
8 rows in set (0.00 sec)

mysql> desc expense;
+---------------+---------------+------+-----+-------------------+----------------+
| Field         | Type          | Null | Key | Default           | Extra          |
+---------------+---------------+------+-----+-------------------+----------------+
| expenseId     | int(11)       | NO   | PRI | NULL              | auto_increment |
| expenseAmount | decimal(10,2) | NO   |     | NULL              |                |
| spentOnDate   | datetime      | NO   |     | NULL              |                |
| note          | varchar(25)   | YES  |     | NULL              |                |
| userId        | int(11)       | NO   | MUL | NULL              |                |
| createdOn     | datetime      | YES  |     | CURRENT_TIMESTAMP |                |
| deletedOn     | datetime      | YES  |     | NULL              |                |
+---------------+---------------+------+-----+-------------------+----------------+
7 rows in set (0.00 sec)
```

DDL

SCHEMA : salary_manager

TABLES					
- User Block - User - UserInfo - UserAddress - UserLog	- Transaction block - Transaction - Income_Transaction - Expense_Transaction	- Earning block - Job - Income	- Expense block - Expense	- Independent - Address	

Queries

```
CREATE TABLE address (
  addressId INT(11) NOT NULL AUTO_INCREMENT,
  street VARCHAR(25) NULL DEFAULT NULL,
  city VARCHAR(20) NULL DEFAULT NULL,
  state CHAR(2) NULL DEFAULT NULL,
  zipcode INT(11) NULL DEFAULT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (addressId));

CREATE TABLE user (
  userId INT(11) NOT NULL AUTO_INCREMENT,
  username VARCHAR(10) NOT NULL,
  password VARCHAR(10) NOT NULL,
  recoveryAnswer VARCHAR(45) NULL DEFAULT NULL,
  recoveryEmail VARCHAR(30) NULL DEFAULT NULL,
  createdOn DATETIME NULL DEFAULT NULL,
  lastUpdatedOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (userId));

CREATE TABLE expense (
  expenseId INT(11) NOT NULL AUTO_INCREMENT,
  expenseAmount DECIMAL(10,2) NOT NULL,
  spentOnDate DATETIME NOT NULL,
  note VARCHAR(25) NULL DEFAULT NULL,
  userId INT(11) NOT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (expenseId),
  CONSTRAINT expenseUserFore
    FOREIGN KEY (userId)
    REFERENCES user (userId));



CREATE TABLE transaction (
  transactionId INT(11) NOT NULL AUTO_INCREMENT,
  netAmount DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT '0.00',
  asOnDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  userId INT(11) NOT NULL,
  isIncome CHAR(1) NULL DEFAULT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (transactionId),
  CONSTRAINT userTransaction
```

```
CREATE TABLE income (
  incomeId INT(11) NOT NULL AUTO_INCREMENT,
  incomeAmount DECIMAL(10,2) NOT NULL,
  earnedDate DATETIME NOT NULL,
  note VARCHAR(25) NULL DEFAULT NULL,
  jobId INT(11) NOT NULL,
  userId INT(11) NOT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (incomeId),
  CONSTRAINT jobIncomeForeg
    FOREIGN KEY (jobId)
    REFERENCES job (jobId));

CREATE TABLE income_transaction (
  incomeTransactionId INT(11) NOT NULL AUTO_INCREMENT,
  incomeId INT(11) NOT NULL,
  transactionId INT(11) NOT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (incomeTransactionId),
  CONSTRAINT incomeIncomeTxnF
    FOREIGN KEY (incomeId)
    REFERENCES income (incomeId)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT transactionIncomeTxnF
    FOREIGN KEY (transactionId)
    REFERENCES transaction (transactionId));


CREATE TABLE user_address (
  userAddressId INT(11) NOT NULL AUTO_INCREMENT,
  userId INT(11) NOT NULL,
  addressId INT(11) NOT NULL,
  PRIMARY KEY (userAddressId),
  CONSTRAINT addressUserFore
    FOREIGN KEY (addressId)
    REFERENCES address (addressId)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT userIdAddFore
    FOREIGN KEY (userId)
    REFERENCES user (userId));
```

```
    FOREIGN KEY (userId)
    REFERENCES user (userId));

CREATE TABLE expense_transaction (
  expenseTransactionId INT(11) NOT NULL AUTO_INCREMENT,
  expenseId INT(11) NOT NULL,
  transactionId INT(11) NOT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (expenseTransactionId),
  CONSTRAINT expenseForeign
    FOREIGN KEY (expenseId)
    REFERENCES expense (expenseId)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT transactionForeign
    FOREIGN KEY (transactionId)
    REFERENCES transaction (transactionId));


CREATE TABLE job (
  jobId INT(11) NOT NULL AUTO_INCREMENT,
  roleName VARCHAR(25) NOT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (jobId));
```

```
CREATE TABLE userinfo (
  userInfoId INT(11) NOT NULL AUTO_INCREMENT,
  userId INT(11) NOT NULL,
  firstName VARCHAR(25) NULL DEFAULT NULL,
  middleName VARCHAR(25) NULL DEFAULT NULL,
  lastName VARCHAR(25) NULL DEFAULT NULL,
  phoneNumber BIGINT(20) NULL DEFAULT NULL,
  createdOn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (userInfoId),
  CONSTRAINT userInfoForeign
    FOREIGN KEY (userId)
    REFERENCES user (userId));

CREATE TABLE userlog (
  userLogId INT(11) NOT NULL AUTO_INCREMENT,
  userId INT(11) NOT NULL,
  lastLoggedIn DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  lastLoggedOut DATETIME NULL DEFAULT NULL,
  deletedOn DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (userLogId),
  CONSTRAINT userLogIDForeign
    FOREIGN KEY (userId)
    REFERENCES user (userId));
```

DML

Expense Transaction

++++++++ EXPENSE + TRANSACTION ++++++++++++
tables:
- income
- income_transaction
- transaction

```
# <!--- START initial set up ---!>
# step 1: ran initially only once

# trigger setup to catch recently inserted expense amount
set @recent_expense_amount=0.0;
create trigger get_recent_expense_amount AFTER INSERT ON expense for each row set @recent_expense_amount = NEW.expenseAmount;

# trigger to capture current userId who inserted the expense
set @current_expense_userId=0;
create trigger get_current_expense_userId AFTER INSERT ON expense for each row set @current_expense_userId = NEW.userId;

# trigger to store the recently created expenseId
set @recent_expenseId=0;
create trigger get_recent_expenseId AFTER INSERT ON expense for each row set @recent_expenseId = NEW.expenseId;

# this trigger is commonly used by income and expense entities
# trigger to capture recently created transaction id
set @recent_transactionId=0;
create trigger get_recent_transactionId AFTER INSERT ON transaction for each row set @recent_transactionId = NEW.transactionId;
# <!--- END initial set up ---!>

# steps 2,3,4 and 5 will definitely executes for all income transactions

# step 2: EXPENSE table insertion
```

```
# Note: if expense amount is greater than the net income amount of the user, the followings will not be executed. even if the followings executes some whow by mistake, the transaction.netAmount will not accept negative entries thus erroring out (ERROR 1264 (22003): Out of range value for column 'netAmount' at row 1).
insert into expense(expenseAmount, spentOnDate, note, userId) values(680, NOW(), 'Nutella and bread', 7);

# step 3: to retrive recent netAmount available at TRANSACTION.netAmount of logged in user.
set @recent_net_amount_by_userId = if((select exists(select 1 from transaction where userId = @current_expense_userId limit 1)) !=1,0.0,(select netAmount from transaction where userId = @current_expense_userId order by transactionId DESC limit 1));

# step 4: TRANSACTION table insertion
insert into transaction(netAmount, userId, isIncome) values((@recent_net_amount_by_userId - @recent_expense_amount), @current_expense_userId, 'N');

# step 5: INCOME_TRANSACTION table insertion
insert into expense_transaction(expenseId, transactionId) values(@recent_expenseId, @recent_transactionId);

# step 6: check the table values queries (to verify data integrity)
select * from expense;
select * from transaction;
select * from expense_transaction;
select @recent_expense_amount, @recent_expenseId, @current_expense_userId, @recent_net_amount_by_userId;


# sample data to test - whenever this done, steps: 3, 4 & 5 definitely to be run after this income data insertion;
insert into expense(expenseAmount, spentOnDate, note, userId) values(680, '2017-11-20', 'Nutella and bread', 7);
insert into expense(expenseAmount, spentOnDate, note, userId) values(300, '2017-11-21', 'gas', 8);
insert into expense(expenseAmount, spentOnDate, note, userId) values(1340, '2017-11-22', 'charity', 7);
insert into expense(expenseAmount, spentOnDate, note, userId) values(68.34, '2017-11-22', 'tshirts', 7);
insert into expense(expenseAmount, spentOnDate, note, userId) values(12.56, '2017-11-24', 'eggs', 8);
```

Income Transaction

```
++++++++ INCOME + TRANSACTION ++++++++++++
tables:
- income
- income_transaction
- transaction

# <!--- START initial set up ---!>
# step 1: ran initially only once

# trigger setup to catch recently inserted income amount
set @recent_income_amount=0.0;
create trigger get_recent_income_amount AFTER INSERT ON income for each row set @recent_income_amount = NEW.incomeAmount;

# trigger to capture current userId who inserted the income
set @current_income_userId=0;
create trigger get_current_income_userId AFTER INSERT ON income for each row set @current_income_userId = NEW.userId;

# trigger to store the recently created incomeId
set @recent_incomeId=0;
create trigger get_recent_incomeId AFTER INSERT ON INCOME for each row set @recent_incomeId = NEW.incomeId;
# <!--- END initial set up ---!>

# steps 2,3,4 and 5 will definitely executes for all income transactions

# step 2: INCOME table insertion
insert into income(incomeAmount, earnedDate, note, jobRole, userId) values(3500, '2017-11-2', 'job at intern soft', 'Software Engineer Intern', 7);
# step 3: to retrive recent netAmount available at TRANSACTION.netAmount of logged in user.
set @recent_net_amount_by_userId = if( (select exists(select 1 from transaction where userId = @current_income_userId limit 1)) !=1,0.0,(select netAmount from transaction where userId = @current_income_userId order by transactionId DESC limit 1) );

# step 4: TRANSACTION table insertion
insert into transaction(netAmount, userId, isIncome) values((@recent_income_amount + @recent_net_amount_by_userId), @current_income_userId, 'Y');

# step 5: INCOME_TRANSACTION table insertion
insert into income_transaction(incomeId, transactionId) values(@recent_incomeId, @recent_transactionId);


# step 6: check the table values queries (to verify data integrity)
select * from income;
```

```
select * from expense;
select * from transaction;
select * from income_transaction;
select * from expense_transaction;
select @recent_income_amount, @recent_incomeId, @current_income_userId, @recent_net_amount_by_userId;
```

Transaction SQL

```
################
Retrieving income and expense from transaction table
>>>FOR INCOME<<<<
mysql> select * from income limit 1;
+----------+--------------+------------+--------+-------------------------+--------+---------------------+-----------+
| incomeId | incomeAmount | earnedDate | note   | jobRole                 | userId | createdOn           | deletedOn |
+----------+--------------+------------+--------+-------------------------+--------+---------------------+-----------+
|       12 |      3500.78 | 2017-11-03 | Intern | Software Engineer Intern |      4 | 2017-12-06 15:29:39 | NULL      |
+----------+--------------+------------+--------+-------------------------+--------+---------------------+-----------+
1 row in set (0.00 sec)

mysql> select * from income_transaction limit 1;
+---------------------+----------+---------------+---------------------+-----------+
| incomeTransactionId | incomeId | transactionId | createdOn           | deletedOn |
+---------------------+----------+---------------+---------------------+-----------+
|                   6 |       12 |            11 | 2017-12-06 15:29:39 | NULL      |
+---------------------+----------+---------------+---------------------+-----------+
1 row in set (0.00 sec)

mysql> select * from transaction limit 1;
+---------------+-----------+---------------------+--------+----------+---------------------+-----------+
| transactionId | netAmount | asOnDate            | userId | isIncome | createdOn           | deletedOn |
+---------------+-----------+---------------------+--------+----------+---------------------+-----------+
|            11 |   3500.78 | 2017-12-06 15:29:39 |      4 | Y        | 2017-12-06 15:29:39 | NULL      |
+---------------+-----------+---------------------+--------+----------+---------------------+-----------+
1 row in set (0.00 sec)

mysql> select income.incomeAmount, income.earnedDate, income.note, income.jobRole from income where incomeId = (select
income_transaction.incomeId from income_transaction where transactionId = 11) and income.deletedOn is null;
+--------------+------------+--------+-------------------------+
| incomeAmount | earnedDate | note   | jobRole                 |
+--------------+------------+--------+-------------------------+
|      3500.78 | 2017-11-03 | Intern | Software Engineer Intern |
+--------------+------------+--------+-------------------------+
1 row in set (0.02 sec)


>>> FOR EXPENSE <<<
mysql> select * from expense limit 1;
+-----------+---------------+-------------+------+--------+---------------------+-----------+
| expenseId | expenseAmount | spentOnDate | note | userId | createdOn           | deletedOn |
+-----------+---------------+-------------+------+--------+---------------------+-----------+
|         5 |        300.00 | 2017-11-05  | Rent |      4 | 2017-12-06 15:30:43 | NULL      |
+-----------+---------------+-------------+------+--------+---------------------+-----------+
1 row in set (0.00 sec)

mysql> select * from expense_transaction limit 1;
+----------------------+-----------+---------------+---------------------+-----------+
| expenseTransactionId | expenseId | transactionId | createdOn           | deletedOn |
+----------------------+-----------+---------------+---------------------+-----------+
|                    4 |         5 |            12 | 2017-12-06 15:30:44 | NULL      |
+----------------------+-----------+---------------+---------------------+-----------+
1 row in set (0.00 sec)

mysql> select * from transaction where transactionId=12;
+---------------+-----------+---------------------+--------+----------+---------------------+-----------+
| transactionId | netAmount | asOnDate            | userId | isIncome | createdOn           | deletedOn |
+---------------+-----------+---------------------+--------+----------+---------------------+-----------+
|            12 |   3200.78 | 2017-12-06 15:30:44 |      4 | N        | 2017-12-06 15:30:44 | NULL      |
+---------------+-----------+---------------------+--------+----------+---------------------+-----------+
1 row in set (0.00 sec)
```

```
mysql> select expense.expenseAmount, expense.note, expense.spentOnDate from expense where expense.expenseId = (select expense_transaction.expenseId from expense_transaction where transactionId = 12) and expense.deletedOn is null;
+---------------+------+-------------+
| expenseAmount | note | spentOnDate |
+---------------+------+-------------+
|        300.00 | Rent | 2017-11-05  |
+---------------+------+-------------+
1 row in set (0.00 sec)
```

Net Amount Calculation:

```
mysql> select sum(incomeAmount) from income where userId=4;
+-------------------+
| sum(incomeAmount) |
+-------------------+
|          10952.28 |
+-------------------+
1 row in set (0.00 sec)

mysql> select sum(expenseAmount) from expense where userId=4;
+--------------------+
| sum(expenseAmount) |
+--------------------+
|             370.22 |
+--------------------+
1 row in set (0.00 sec)

mysql> select sum(incomeAmount) - (select sum(expenseAmount) from expense where userId=4) from income where userId=4;
+-----------------------------------------------------------------------------+
| sum(incomeAmount) - (select sum(expenseAmount) from expense where userId=4) |
+-----------------------------------------------------------------------------+
|                                                                    10582.06 |
+-----------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

ALSo, the last row of the transaction table of appropriate userId will hold net amount value;

```
mysql> select netAmount from transaction where userId = 4 order by createdOn desc limit 1;
+-----------+
| netAmount |
+-----------+
|  10582.06 |
+-----------+
1 row in set (0.00 sec)
```

User table related queries

+++++++++ INDEPENDENT ++++++++++
tables:
- address

Action:

```
mysql> insert into address(street, city, state, zipcode) values('45, Vern Terr', 'Poughkeepsie', 'NY', 12601), ('23, Gifford Ave', 'Sansy', 'CA', 929203), ('Main Street', 'Albany', 'NY', 12601), ('George Wash Street', 'BridgePort', 'NY', 12601);
```

+++++++++ USER ++++++++++
tables:
- user
- userInfo
- userAddress

Actions:
- USER TABLE

```
mysql> insert into user(username, password, recoveryAnswer, recoveryEmail) value ('kev22', 'keviv', 'iavsh', 'techengineervivek@gmail.com'), ('shar37', 'sharan', 'forgot my password', 'shar@yahoo.com'), ('vaish47', 'haiav', 'i never forget anything', 'vaish@gmail.com');
```

- USERINFO TABLE

mysql> insert into userInfo(userId, firstName, middleName, lastName, phoneNumber) values(6, 'Vivek', 'V', 'Surulimuthu', 9292946551), (7, 'Sharan', 'Junior', 'Keviv', 9994451794), (8, 'Vaish', '', 'Leela', 9566133013);

- USERADDRESS TABLE
mysql> insert into user_address(userId, addressId) values(6,5), (7,7), (8,6);

- USERLOG TABLE
for logIn:
#setup of trigger done to catch Current UserLogged in!
# trigger to store recently created userlogId
set @recent_userLogId=0;
set @recent_loggedIn_userId=0;
create trigger get_recent_userLogId AFTER INSERT ON UserLog for each row set @recent_userLogId = NEW.userLogId;
create trigger get_recent_loggedUserId AFTER INSERT ON UserLog for each row set @recent_loggedIn_userId = NEW.userId;

@recent_userLogId = (select userLogId from userlog order by userLogId desc limit 1);

insert into userlog(userId, lastLoggedIn) values(4, NOW());

for logout:
@recent_userLogId can be obtained from trigger or by the following query too. (have to check the efficiecny)
set @recent_userLogId = (select userLogId from userlog order by userLogId desc limit 1)
update userLog set lastLoggedOut= NOW() where userLogId=@recent_userLogId

Reason:
- to map the difference between user log in and log out status so that we can analyze the active usage time spent by a user.

- USERADDRESS TABLE
mysql> select * from user;
+--------+----------+----------+------------------------+----------------------------+---------------------+---------------------+-----------+
| userId | username | password | recoveryAnswer | recoveryEmail | createdOn | lastUpdatedOn | deletedOn |
+--------+----------+----------+------------------------+----------------------------+---------------------+---------------------+-----------+
4	kev22	keviv	iavsh	techengineervivek@gmail.com	2017-12-05 23:00:13	2017-12-05 23:00:13	NULL
5	shar37	sharan	forgot my password	shar@yahoo.com	2017-12-05 23:00:13	2017-12-05 23:00:13	NULL
6	vaish47	haiav	i never forget anything	vaish@gmail.com	2017-12-05 23:00:13	2017-12-05 23:00:13	NULL
+--------+----------+----------+------------------------+----------------------------+---------------------+---------------------+-----------+
3 rows in set (0.00 sec)

mysql> select * from address;
+-----------+--------------------+--------------+-------+---------+---------------------+-----------+
| addressId | street | city | state | zipcode | createdOn | deletedOn |
+-----------+--------------------+--------------+-------+---------+---------------------+-----------+
1	45, Vern Terr	Poughkeepsie	NY	12601	2017-12-05 22:57:23	NULL
2	23, Gifford Ave	Sansy	CA	929203	2017-12-05 22:57:23	NULL
3	Main Street	Albany	NY	12601	2017-12-05 22:57:23	NULL
4	George Wash Street	BridgePort	NY	12601	2017-12-05 22:57:23	NULL
+-----------+--------------------+--------------+-------+---------+---------------------+-----------+
4 rows in set (0.06 sec)

mysql> select street, city, state, zipcode from address where addressId = (select addressId from user_address where userId = 4);
+---------------+--------------+-------+---------+
| street | city | state | zipcode |
+---------------+--------------+-------+---------+
| 45, Vern Terr | Poughkeepsie | NY | 12601 |
+---------------+--------------+-------+---------+
1 row in set (0.02 sec)

mysql> select concat(street, ', ', state, ', ', zipcode) as addressUser from address where addressId = (select addressId from user_address where userId= 4);
+--------------------------+
| addressUser |
+--------------------------+
| 45, Vern Terr, NY, 12601 |
+--------------------------+
1 row in set (0.00 sec)