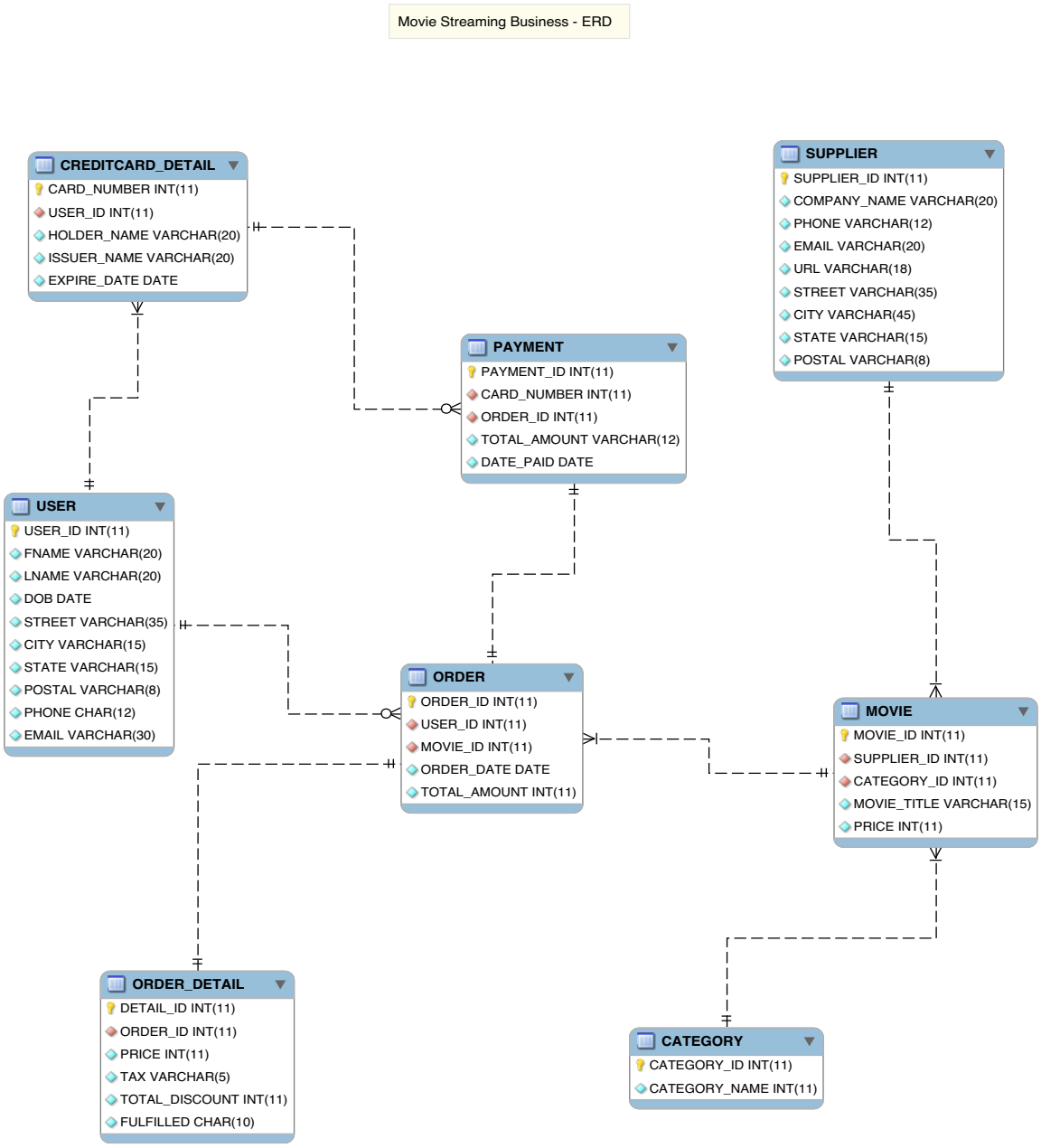
MYSQL Project

This project is based on the database design and management of a movie streaming business.



BUILD_CODE

```
-- -----------------------------------------------------
-- Table `main`.`CATEGORY`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`CATEGORY` (
  `CATEGORY_ID` INT NOT NULL,
  `CATEGORY_NAME` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`CATEGORY_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- -----------------------------------------------------
-- Table `main`.`USER`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`USER` (
  `USER_ID` INT NOT NULL,
  `FNAME` VARCHAR(20) NOT NULL,
  `LNAME` VARCHAR(20) NOT NULL,
  `DOB` DATE NOT NULL,
  `STREET` VARCHAR(35) NOT NULL,
  `CITY` VARCHAR(25) NOT NULL,
  `STATE` VARCHAR(15) NOT NULL,
  `PHONE` CHAR(12) NOT NULL,
  `EMAIL` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`USER_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- -----------------------------------------------------
-- Table `main`.`CREDITCARD`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`CREDITCARD` (
  `CARD_NUMBER` VARCHAR(20) NOT NULL,
  `USER_ID` INT NOT NULL,
  `HOLDER_NAME` VARCHAR(30) NOT NULL,
  `ISSUER_NAME` VARCHAR(20) NOT NULL,
  `EXPIRE_DATE` DATE NOT NULL,
  PRIMARY KEY (`CARD_NUMBER`),
  FOREIGN KEY (`USER_ID`) REFERENCES `main`.`USER` (`USER_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- -----------------------------------------------------
-- Table `main`.`SUPPLIER`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`SUPPLIER` (
  `SUPPLIER_ID` INT NOT NULL,
```

```
  `COMPANY_NAME` VARCHAR(20) NOT NULL,
  `PHONE` VARCHAR(12) NOT NULL,
  `EMAIL` VARCHAR(40) NOT NULL,
  `URL` VARCHAR(40) NOT NULL,
  `STREET` VARCHAR(35) NOT NULL,
  `CITY` VARCHAR(15) NOT NULL,
  `STATE` VARCHAR(15) NOT NULL,
  `POSTAL` VARCHAR(8) NOT NULL,
  PRIMARY KEY (`SUPPLIER_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- -----------------------------------------------------
-- Table `main`.`MOVIE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`MOVIE` (
  `MOVIE_ID` INT NOT NULL,
  `SUPPLIER_ID` INT NOT NULL,
  `MOVIE_TITLE` VARCHAR(50) NOT NULL,
  `CATEGORY_ID` INT NOT NULL,
  `PRICE` INT NOT NULL,
  PRIMARY KEY (`MOVIE_ID`),
  FOREIGN KEY (`SUPPLIER_ID`) REFERENCES `main`.`SUPPLIER` (`SUPPLIER_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (`CATEGORY_ID`) REFERENCES `main`.`CATEGORY` (`CATEGORY_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- -----------------------------------------------------
-- Table `main`.`ORDER`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`ORDERS` (
  `ORDER_ID` INT NOT NULL,
  `USER_ID` INT NOT NULL,
  `MOVIE_ID` INT NOT NULL,
  `ORDER_DATE` DATE NOT NULL,
  `TOTAL_AMOUNT` INT NOT NULL,
  PRIMARY KEY (`ORDER_ID`),
  FOREIGN KEY (`USER_ID`) REFERENCES `main`.`USER` (`USER_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (`MOVIE_ID`) REFERENCES `main`.`MOVIE` (`MOVIE_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- -----------------------------------------------------
-- Table `main`.`PAYMENT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`PAYMENT` (
  `PAYMENT_ID` INT NOT NULL,
  `CARD_NUMBER` VARCHAR(20) NOT NULL,
  `ORDER_ID` INT NOT NULL,
  `TOTAL_AMOUNT` VARCHAR(12) NOT NULL,
  `DATE_PAID` DATE NOT NULL,
  PRIMARY KEY (`PAYMENT_ID`),
  FOREIGN KEY (`CARD_NUMBER`) REFERENCES `main`.`CREDITCARD` (`CARD_NUMBER`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (`ORDER_ID`) REFERENCES `main`.`ORDERS` (`ORDER_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

-- -----------------------------------------------------
-- Table `main`.`ORDER_DETAIL`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `main`.`ORDER_DETAIL` (
  `DETAIL_ID` INT NOT NULL,
  `ORDER_ID` INT NOT NULL,
  `MOVIE_ID` INT NOT NULL,
  `PRICE` INT NOT NULL,
  `TAX` INT NOT NULL,
  `TOTAL_DISCOUNT` INT NOT NULL,
  `FULFILLED` CHAR(10) NOT NULL,
  PRIMARY KEY (`DETAIL_ID`),
  FOREIGN KEY (`ORDER_ID`) REFERENCES `main`.`ORDERS` (`ORDER_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (`MOVIE_ID`) REFERENCES `main`.`MOVIE` (`MOVIE_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

LOAD_CODE

```
-- -----------------------------------------------------
-- Load data in the database
-- -----------------------------------------------------
LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/category.csv'
INTO TABLE CATEGORY
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/user.csv'
INTO TABLE USER
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/creditcard.csv'
INTO TABLE CREDITCARD
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/supplier.csv'
INTO TABLE SUPPLIER
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/movie.csv'
INTO TABLE MOVIE
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/order.csv'
INTO TABLE ORDERS
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/payment.csv'
INTO TABLE PAYMENT
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE '/Users/slick/Documents/MYSQL_Project/order_detail.csv'
INTO TABLE ORDER_DETAIL
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

QUERIES

```
-- ------------------------------------------------------------------------
-- Total movie sales for 2018
-- ------------------------------------------------------------------------
SELECT ORDERS.MOVIE_ID, MOVIE_TITLE, SUM(TOTAL_AMOUNT) AS SALES
FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
WHERE ORDER_DATE IN (SELECT ORDER_DATE FROM ORDERS WHERE ORDER_DATE BETWEEN '2018-01-01' AND '2019-
01-01')
GROUP BY ORDERS.MOVIE_ID;

-- ------------------------------------------------------------------------
-- Share of total sales by state
-- ------------------------------------------------------------------------
WITH
Q1 AS (SELECT STATE, COUNT(ORDER_ID) AS STATE_SALES FROM ORDERS INNER JOIN USER ON ORDERS.USER_ID =
USER.USER_ID GROUP BY STATE ORDER BY STATE),
Q2 AS (SELECT COUNT(*) AS TOTAL_SALES FROM ORDERS)
SELECT Q1.STATE, ROUND((100 * (Q1.STATE_SALES/Q2.TOTAL_SALES)),2) AS SHARE_OF_TOTAL_SALES
FROM Q1 CROSS JOIN Q2;

-- ------------------------------------------------------------------------
-- Sales by state in units & dollars for the first quarter
-- ------------------------------------------------------------------------
SELECT STATE, SUM(TOTAL_AMOUNT) AS $$_SALES, COUNT(ORDER_ID) AS UNIT_SALES
FROM USER INNER JOIN ORDERS ON USER.USER_ID = ORDERS.USER_ID
WHERE ORDER_DATE BETWEEN '2019-01-01' AND '2019-03-30'
GROUP BY STATE;

-- ------------------------------------------------------------------------
-- Users with transactions above average
-- ------------------------------------------------------------------------
SELECT USER.USER_ID, TIMESTAMPDIFF(YEAR, DOB, CURDATE()) AS AGE, STATE
FROM USER INNER JOIN ORDERS ON USER.USER_ID = ORDERS.USER_ID
WHERE TOTAL_AMOUNT > (SELECT AVG(TOTAL_AMOUNT) AS AVG_SALES
FROM ORDERS);

-- ------------------------------------------------------------------------
-- Average discount amount per order in CA, WA, FL, NY
-- ------------------------------------------------------------------------
SELECT STATE, AVG(TOTAL_DISCOUNT) AS AVG_DISCOUNT
FROM ORDER_DETAIL INNER JOIN ORDERS ON ORDER_DETAIL.ORDER_ID = ORDERS.ORDER_ID
INNER JOIN USER ON ORDERS.USER_ID = USER.USER_ID
WHERE STATE IN ('CA', 'WA', 'FL', 'NY')
GROUP BY STATE
ORDER BY AVG_DISCOUNT DESC;

-- ------------------------------------------------------------------------
-- Suppliers of the most expensive movie streams
-- ------------------------------------------------------------------------
SELECT COMPANY_NAME, MOVIE_TITLE, PRICE
FROM MOVIE INNER JOIN SUPPLIER ON MOVIE.SUPPLIER_ID = SUPPLIER.SUPPLIER_ID
WHERE PRICE = (SELECT MAX(PRICE) FROM MOVIE);
```

```
-- ---------------------------------------------------------------------------
-- Average movie price per supplier & category
-- ---------------------------------------------------------------------------
SELECT COMPANY_NAME, CATEGORY_NAME, ROUND(AVG(PRICE),2) AS AVG_PRICE
FROM CATEGORY INNER JOIN MOVIE ON CATEGORY.CATEGORY_ID = MOVIE.CATEGORY_ID
INNER JOIN SUPPLIER ON MOVIE.SUPPLIER_ID = SUPPLIER.SUPPLIER_ID
GROUP BY 1, 2
ORDER BY 1 DESC;

-- ---------------------------------------------------------------------------
-- Most watched categories
-- ---------------------------------------------------------------------------
SELECT CATEGORY_NAME, COUNT(ORDER_ID) AS UNITS_SOLD
FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
INNER JOIN CATEGORY ON  MOVIE.CATEGORY_ID = CATEGORY.CATEGORY_ID
GROUP BY CATEGORY_NAME
ORDER BY UNITS_SOLD DESC;

-- ---------------------------------------------------------------------------
-- Top Suppliers for the company
-- ---------------------------------------------------------------------------
SELECT COMPANY_NAME, COUNT(MOVIE_ID) AS MOVIES_SUPPLIED
FROM MOVIE INNER JOIN SUPPLIER ON MOVIE.SUPPLIER_ID = SUPPLIER.SUPPLIER_ID
GROUP BY 1
HAVING MOVIES_SUPPLIED > 5
ORDER BY 1
LIMIT 4;

-- ---------------------------------------------------------------------------
-- Dollar and unit sales by movie
-- ---------------------------------------------------------------------------
SELECT MOVIE_TITLE, SUM(TOTAL_AMOUNT) AS $$_SALES, COUNT(ORDERS.MOVIE_ID) AS UNIT_SALES
FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
GROUP BY MOVIE_TITLE
ORDER BY SUM(TOTAL_AMOUNT) DESC;

-- ---------------------------------------------------------------------------
-- Total sales by a specific movie 'Mr. & Mrs. Smith'
-- ---------------------------------------------------------------------------
SELECT MOVIE_TITLE, SUM(TOTAL_AMOUNT) AS SALES
FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
WHERE MOVIE_TITLE LIKE '%Smith%'
GROUP BY MOVIE_TITLE;

-- ---------------------------------------------------------------------------
-- Top 5 categories by average price
-- ---------------------------------------------------------------------------
SELECT CATEGORY_NAME,ROUND(AVG(PRICE),2)
FROM MOVIE INNER JOIN CATEGORY ON MOVIE.CATEGORY_ID = CATEGORY.CATEGORY_ID
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

LOGFILE

```
Last login: Mon Nov 11 23:42:42 on ttys000
slick@Victors-Air ~ % /usr/local/mysql/bin/mysql -u Slick -p --local-infile
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 26
Server version: 8.0.18 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SELECT VERSION(), CURRENT_DATE;
+-----------+--------------+
| VERSION() | CURRENT_DATE |
+-----------+--------------+
| 8.0.18    | 2019-11-11   |
+-----------+--------------+
1 row in set (0.00 sec)

mysql> SELECT USER();
+-----------------+
| USER()          |
+-----------------+
| Slick@localhost |
+-----------------+
1 row in set (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| main               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> USE main;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+---------------+
| Tables_in_main |
+---------------+
| CATEGORY       |
| CREDITCARD     |
```

```
| MOVIE          |
| ORDER_DETAIL   |
| ORDERS         |
| PAYMENT        |
| SUPPLIER       |
| USER           |
+----------------+
8 rows in set (0.00 sec)

mysql> -- -------------------------------------------------
mysql> -- Total movie sales for 2018
mysql> -- -------------------------------------------------
mysql> SELECT ORDERS.MOVIE_ID, MOVIE_TITLE, SUM(TOTAL_AMOUNT) AS SALES
    -> FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
    -> WHERE ORDER_DATE IN (SELECT ORDER_DATE FROM ORDERS WHERE ORDER_DATE
BETWEEN '2018-01-01' AND '2019-01-01')
    -> GROUP BY ORDERS.MOVIE_ID;
+-----------+-------------------------------------------+-------+
| MOVIE_ID  | MOVIE_TITLE                               | SALES |
+-----------+-------------------------------------------+-------+
|   6201176 | Tremors 4                                 |   117 |
| 299924201 | Stolen Face                               |   113 |
| 750964275 | Do the Right Thing                        |   110 |
|   2730662 | Obsession                                 |   116 |
| 995073172 | Cain and Mabel                            |   111 |
| 662977821 | The Expedition to the End of the World    |   114 |
| 498286688 | Pushover                                  |   112 |
| 952739038 | "Never Play Clever Again"                 |   220 |
| 788553520 | Hyde Park on Hudson                       |   214 |
| 160322796 | Mr. & Mrs. Smith                          |   119 |
| 850222065 | Craig Ferguson: Does This Need to Be Said? |   311 |
| 828408796 | Hostage                                   |   266 |
|  45140155 | Verboten                                  |   211 |
| 380055414 | Damage                                    |   219 |
| 996062671 | Riverworld                                |   113 |
| 438373946 | Played                                    |   113 |
| 982452201 | New Waterford Girl                        |   317 |
| 129839421 | "Woman in Red"                            |   211 |
| 252159579 | Adrift                                    |   113 |
| 650291372 | "Devil's Ground"                          |   216 |
| 576093111 | 187                                       |   116 |
| 262690274 | "Stranger's Heart"                        |   111 |
| 893140864 | One Magic Christmas                       |   110 |
| 159722467 | "Arena"                                   |   213 |
| 868172488 | Torrid Zone                               |   318 |
| 165107385 | Rahtree: Flower of the Night              |   419 |
| 785217312 | Noise                                     |   214 |
| 223617299 | Tupac: Resurrection                       |   316 |
| 215675495 | "Charming Mass Suicide"                   |   116 |
| 904956039 | "Ghost Ship"                              |   218 |
| 781222840 | Carlito's Way                             |   217 |
| 911082249 | Anchors Aweigh                            |   113 |
| 995068617 | Soundbreaker                              |   119 |
+-----------+-------------------------------------------+-------+
41 rows in set (0.00 sec)
```

```
mysql> -- ------------------------------------------------
mysql> -- Share of total sales by state
mysql> -- ------------------------------------------------
mysql> WITH
    -> Q1 AS (SELECT STATE, COUNT(ORDER_ID) AS STATE_SALES FROM ORDERS INNER JOIN
USER ON ORDERS.USER_ID = USER.USER_ID GROUP BY STATE ORDER BY STATE),
    -> Q2 AS (SELECT COUNT(*) AS TOTAL_SALES FROM ORDERS)
    -> SELECT Q1.STATE, ROUND((100 * (Q1.STATE_SALES/Q2.TOTAL_SALES)),2) AS
SHARE_OF_TOTAL_SALES
    -> FROM Q1 CROSS JOIN Q2;
+-------+---------------------+
| STATE | SHARE_OF_TOTAL_SALES |
+-------+---------------------+
| AL    |                3.00 |
| AR    |                1.00 |
| AZ    |                3.00 |
| CA    |               10.00 |
| CO    |                3.00 |
| DC    |                1.00 |
| FL    |                3.00 |
| GA    |               26.00 |
| HI    |                1.00 |
| ID    |                1.00 |
| IL    |                1.00 |
| IN    |                1.00 |
| LA    |                2.00 |
| NE    |                1.00 |
| NJ    |                2.00 |
| NM    |                1.00 |
| NV    |                1.00 |
| NY    |                7.00 |
| OK    |               17.00 |
| OR    |                1.00 |
| PA    |                3.00 |
| SD    |                1.00 |
| TN    |                1.00 |
| TX    |                3.00 |
| VA    |                1.00 |
| WA    |                3.00 |
| WI    |                2.00 |
+-------+---------------------+
31 rows in set (0.00 sec)

mysql> -- ------------------------------------------------
mysql> -- Sales by state for the first quarter
mysql> -- ------------------------------------------------
mysql> SELECT STATE, SUM(TOTAL_AMOUNT) AS $$_SALES, COUNT(ORDER_ID) AS UNIT_SALES
    -> FROM USER INNER JOIN ORDERS ON USER.USER_ID = ORDERS.USER_ID
    -> WHERE ORDER_DATE BETWEEN '2019-01-01' AND '2019-03-30'
    -> GROUP BY STATE;
+-------+----------+------------+
| STATE | $$_SALES | UNIT_SALES |
+-------+----------+------------+
| GA    |   319.54 |         27 |
| MD    |   114.67 |          8 |
| FL    |   201.32 |         21 |
```

```
| CA    |   127.43 |         12 |
| NM    |   117.55 |         11 |
| WA    |   219.34 |         19 |
| IN    |   111.56 |          7 |
| NY    |   219.77 |         18 |
+-------+----------+------------+
8 rows in set (0.00 sec)

mysql> -- -------------------------------------------------------
mysql> -- Users with transactions above average
mysql> -- -------------------------------------------------------
mysql> SELECT USER.USER_ID, TIMESTAMPDIFF(YEAR, DOB, CURDATE()) AS AGE, STATE
    -> FROM USER INNER JOIN ORDERS ON USER.USER_ID = ORDERS.USER_ID
    -> WHERE TOTAL_AMOUNT > (SELECT AVG(TOTAL_AMOUNT) AS AVG_SALES
    -> FROM ORDERS);
+-----------+------+-------+
| USER_ID   | AGE  | STATE |
+-----------+------+-------+
|  25499262 |   22 | WA    |
| 511307243 |   30 | OK    |
| 731096902 |   33 | NY    |
| 628172565 |   32 | OK    |
| 602847947 |   21 | GA    |
| 600557466 |   37 | FL    |
| 572831602 |   27 | NJ    |
| 395447695 |   28 | WI    |
| 602847947 |   21 | GA    |
| 149279414 |   32 | NY    |
| 648726387 |   33 | NY    |
| 773776502 |   23 | NC    |
| 512729529 |   25 | IL    |
| 602847947 |   21 | GA    |
| 511307243 |   30 | OK    |
| 512137846 |   25 | TX    |
| 223259711 |   32 | FL    |
| 602847947 |   21 | GA    |
| 776930484 |   29 | NC    |
| 602847947 |   21 | GA    |
+-----------+------+-------+
43 rows in set (0.00 sec)

mysql> -- -------------------------------------------------------
mysql> -- Average discount amount per order in CA, WA, FL, NY
mysql> -- -------------------------------------------------------
mysql> SELECT STATE, AVG(TOTAL_DISCOUNT) AS AVG_DISCOUNT
    -> FROM ORDER_DETAIL INNER JOIN ORDERS ON ORDER_DETAIL.ORDER_ID =
ORDERS.ORDER_ID
    -> INNER JOIN USER ON ORDERS.USER_ID = USER.USER_ID
    -> WHERE STATE IN ('CA', 'WA', 'FL', 'NY')
    -> GROUP BY STATE
    -> ORDER BY AVG_DISCOUNT DESC;
+-------+--------------+
| STATE | AVG_DISCOUNT |
+-------+--------------+
| WA    |      17.33   |
| FL    |      14.00   |
```

```
| CA    |      14.00   |
| NY    |      13.14   |
+-------+--------------+
4 rows in set (0.00 sec)

mysql> -- -----------------------------------------------------
mysql> -- Suppliers of the most expensive movie streams
mysql> -- -----------------------------------------------------
mysql> SELECT COMPANY_NAME, MOVIE_TITLE, PRICE
    -> FROM MOVIE INNER JOIN SUPPLIER ON MOVIE.SUPPLIER_ID = SUPPLIER.SUPPLIER_ID
    -> WHERE PRICE = (SELECT MAX(PRICE) FROM MOVIE);
+--------------+---------------------------------------------+-------+
| COMPANY_NAME | MOVIE_TITLE                                 | PRICE |
+--------------+---------------------------------------------+-------+
| Camido       | Spiders: The Diamond Ship                   | 15.00 |
| Skivee       | Copenhagen                                  | 15.00 |
| Kwilith      | In Order of Disappearance                   | 15.00 |
| Camido       | Extreme Ops                                 | 15.00 |
| Camido       | Theatre of Blood                            | 15.00 |
| Skivee       | Craig Ferguson: Does This Need to Be Said?  | 15.00 |
| Skivee       | "Ghost Ship"                                | 15.00 |
| Bluezoom     | Waking Up in Reno                           | 15.00 |
| Skivee       | Scream                                      | 15.00 |
| Skippad      | New Waterford Girl                          | 15.00 |
+--------------+---------------------------------------------+-------+
10 rows in set (0.00 sec)

mysql> -- -----------------------------------------------------
mysql> -- Average movie price per supplier & category
mysql> -- -----------------------------------------------------
mysql> SELECT COMPANY_NAME, CATEGORY_NAME, ROUND(AVG(PRICE),2) AS AVG_PRICE
    -> FROM CATEGORY INNER JOIN MOVIE ON CATEGORY.CATEGORY_ID = MOVIE.CATEGORY_ID
    -> INNER JOIN SUPPLIER ON MOVIE.SUPPLIER_ID = SUPPLIER.SUPPLIER_ID
    -> GROUP BY 1, 2
    -> ORDER BY 1 DESC;
+--------------+---------------------------------------------+-----------+
| COMPANY_NAME | CATEGORY_NAME                               | AVG_PRICE |
+--------------+---------------------------------------------+-----------+
| Yadel        | Crime|Drama                                 |     11.00 |
| Wikizz       | Drama|War                                   |     12.00 |
| Trilia       | Animation|Documentary|War                   |     14.00 |
| Thoughtbridge| Thriller                                    |      9.00 |
| Skyndu       | Comedy|Drama                               |      8.00 |
| Skivee       | Action|Adventure|Comedy|Sci-Fi             |     15.00 |
| Skivee       | Children                                    |     13.00 |
| Skivee       | Documentary                                 |     11.67 |
| Skivee       | Drama                                       |     12.00 |
| Skippad      | Documentary|War                             |     15.00 |
| Skimia       | Comedy                                      |     14.00 |
| Shufflester  | Children|Comedy|Drama|Fantasy               |     12.00 |
| Shufflebeat  | Action|Mystery                             |     12.00 |
| Kwilith      | Action|Adventure|Horror|Sci-Fi|Thriller    |     15.00 |
| Jetpulse     | Drama                                       |     12.00 |
| Fivechat     | Action|Adventure|Drama|Fantasy             |     12.00 |
| Fivechat     | Comedy                                      |     11.00 |
| Fivechat     | Documentary                                 |     13.00 |
```

```
| Fivechat      | Drama                                          |     11.50 |
| Devcast       | Documentary|Drama                               |      8.00 |
| Dazzlesphere  | Comedy                                         |     10.00 |
| Cogidoo       | Comedy                                         |      8.00 |
| Camido        | Action|War                                     |     14.00 |
| Camido        | Comedy                                         |     11.50 |
| Camido        | Documentary                                    |     11.50 |
| Camido        | Drama                                          |     12.14 |
| Camido        | Mystery                                        |      9.00 |
| Buzzdog       | Drama|Horror|Thriller                          |     14.00 |
| Bluezoom      | Action|Comedy|Thriller                        |     13.00 |
| Bluezoom      | Documentary                                    |     12.00 |
| Bluezoom      | Drama                                          |      9.00 |
| Blognation    | Drama|Thriller                                 |     14.00 |
+---------------+------------------------------------------------+-----------+
31 rows in set (0.00 sec)

mysql> -- -------------------------------------------------------
mysql> -- Most watched categories
mysql> -- -------------------------------------------------------
mysql> SELECT CATEGORY_NAME, COUNT(ORDER_ID) AS UNITS_SOLD
    -> FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
    -> INNER JOIN CATEGORY ON  MOVIE.CATEGORY_ID = CATEGORY.CATEGORY_ID
    -> GROUP BY CATEGORY_NAME
    -> ORDER BY UNITS_SOLD DESC;
+--------------------------------------------+------------+
| CATEGORY_NAME                              | UNITS_SOLD |
+--------------------------------------------+------------+
| Children|Comedy                            |        216 |
| Drama                                      |        213 |
| Crime|Film-Noir|Thriller                    |        210 |
| Documentary                                |        180 |
| Comedy                                     |        160 |
| Action|Mystery                             |        111 |
| Thriller                                   |        110 |
| Children                                   |        100 |
| Documentary|War                            |        100 |
| Comedy|Drama                               |        100 |
| Drama|Thriller                             |         91 |
| Action|Thriller                            |         91 |
| Animation|Documentary|War                  |         89 |
| Mystery                                    |         81 |
| Comedy|Romance                             |         79 |
| Documentary|Drama                          |         75 |
| Drama|War                                  |         71 |
| Crime|Drama                               |         61 |
| Animation|Comedy                           |         61 |
| Drama|Romance                             |         61 |
| Western                                    |         41 |
+--------------------------------------------+------------+
35 rows in set (0.00 sec)
```

```
mysql> -- ------------------------------------------------------
mysql> -- Top Suppliers for the company
mysql> -- ------------------------------------------------------
mysql> SELECT COMPANY_NAME, COUNT(MOVIE_ID) AS MOVIES_SUPPLIED
    -> FROM MOVIE INNER JOIN SUPPLIER ON MOVIE.SUPPLIER_ID = SUPPLIER.SUPPLIER_ID
    -> GROUP BY 1
    -> HAVING MOVIES_SUPPLIED > 5
    -> ORDER BY 1
    -> LIMIT 4;
+--------------+-----------------+
| COMPANY_NAME | MOVIES_SUPPLIED |
+--------------+-----------------+
| Bluezoom     |              11 |
| Camido       |              31 |
| Fivechat     |              16 |
| Skivee       |              21 |
+--------------+-----------------+
4 rows in set (0.00 sec)

mysql> -- ------------------------------------------------------
mysql> -- Dollars and units sales by movie
mysql> -- ------------------------------------------------------
mysql> SELECT MOVIE_TITLE, SUM(TOTAL_AMOUNT) AS $$_SALES, COUNT(ORDERS.MOVIE_ID)
AS UNIT_SALES
    -> FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
    -> GROUP BY MOVIE_TITLE
    -> ORDER BY SUM(TOTAL_AMOUNT) DESC;
+-----------------------------------------+----------+------------+
| MOVIE_TITLE                             | $$_SALES | UNIT_SALES |
+-----------------------------------------+----------+------------+
| Mr. & Mrs. Smith                        |   276.76 |         26 |
| Hostage                                 |   248.36 |         20 |
| Spiders: The Diamond Ship               |   200.65 |         18 |
| "Never Play Clever Again"               |   204.66 |         18 |
| Howling III: The Marsupials             |   203.98 |         17 |
| Soundbreaker                            |   198.65 |         17 |
| Eternal Sunshine of the Spotless Mind   |   196.87 |         15 |
| Belizaire the Cajun                     |   193.65 |         15 |
| Italian for Beginners                   |   191.44 |         15 |
| Tales That Witness Madness              |   191.55 |         15 |
| Elevator to the Gallows                 |   139.65 |         15 |
| Rahtree: Flower of the Night            |   129.75 |         14 |
| Damage                                  |   127.11 |         13 |
| Dodes'ka-den (Clickety-Clack)           |   122.55 |         11 |
| Sophie's Revenge                        |   118.65 |         11 |
| "Ghost Ship"                            |   118.11 |         10 |
| Torrid Zone                             |   118.01 |         10 |
| Tremors 4                               |   117.98 |         10 |
| Trick                                   |   117.76 |         10 |
| Carlito's Way                           |   117.45 |         10 |
| New Waterford Girl                      |   117.34 |         11 |
| "Benchwarmers"                          |   117.23 |         10 |
| Quarantine                              |   116.87 |         10 |
| "Charming Mass Suicide"                 |   116.67 |         10 |
| Tupac: Resurrection                     |   116.56 |         10 |
| In Name Only                            |   116.44 |         12 |
```

```
| Copenhagen                                |   114.43 |         14 |
| Noise                                     |   114.34 |         10 |
| In Order of Disappearance                 |   114.29 |         10 |
| Gorgeous                                  |   114.05 |         10 |
| Under the Sun of Satan                    |   114.02 |         10 |
+-------------------------------------------+----------+------------+
46 rows in set (0.00 sec)

mysql> -- ------------------------------------------------------
mysql> -- Total sales by a specific movie 'Mr. & Mrs. Smith'
mysql> -- ------------------------------------------------------
mysql> SELECT MOVIE_TITLE, SUM(TOTAL_AMOUNT) AS SALES
    -> FROM ORDERS INNER JOIN MOVIE ON ORDERS.MOVIE_ID = MOVIE.MOVIE_ID
    -> WHERE MOVIE_TITLE LIKE '%Smith%'
    -> GROUP BY MOVIE_TITLE;
+------------------+-------+
| MOVIE_TITLE      | SALES |
+------------------+-------+
| Mr. & Mrs. Smith |   236 |
+------------------+-------+
1 row in set (0.00 sec)

mysql> -- ------------------------------------------------------
mysql> -- Top 5 categories by average price
mysql> -- ------------------------------------------------------
mysql> SELECT CATEGORY_NAME,ROUND(AVG(PRICE),2) AVG_PRICE
    -> FROM MOVIE INNER JOIN CATEGORY ON MOVIE.CATEGORY_ID = CATEGORY.CATEGORY_ID
    -> GROUP BY 1
    -> ORDER BY 2 DESC
    -> LIMIT 5;
+---------------------------------------+-----------+
| CATEGORY_NAME                         | AVG_PRICE |
+---------------------------------------+-----------+
| Action|Adventure|Horror|Sci-Fi|Thriller |     15.00 |
| Adventure|Animation|Children|Comedy     |     15.00 |
| Action|Adventure|Comedy|Sci-Fi          |     15.00 |
| Documentary|War                       |     15.00 |
| Drama|Horror|Thriller                  |     14.00 |
+---------------------------------------+-----------+
5 rows in set (0.00 sec)
```