GETKICKS

CHILDREN'S SHOES DATABASE

Project by:
Kathleen Lara
Chia Hsin Hsieh
Data Management & SQL
Hult International Business School

Executive Overview

Getkicks is an online store that sells shoes for children from different suppliers. Its business goal is to make sure that they are able to make use of their data to increase in revenue and save on operational costs.

Along with this goal is the **Getkicks database system** that stores data of its customers, orders, inventory, shoe models for children, suppliers and other relevant data collected during the ordering process and the post-shopping experience of the customer.

Some of the key pain points Getkicks is trying to address by building a good relational database system are categorized into three buckets (one of each will have different pain points):

- **Paint Point on Customers:** It's a challenge to understand customers' buying behavior and identify different types of customer groups
- **Paint Point on Customer Retention:** The company is always looking to find ways to improve customer retention, it's difficult to monitor the reviews and net promoter score for every customer
- **Pain Point on Operational Costs Savings:** It's always a challenge to save on operational cost and the company finds it difficult to see which shoes are still in full stock and do not need more supplies

Actionable insights

Getkicks Database System's is made to address the pain points by answering key questions for every pain point that the business has. Its goal is to get insights from key questions that can be formulated from the pain points.

Some of the key questions and insights wished to be derived from these questions are as follows:

Paint Point on Customers: It's important to understand the customers' buying behavior so the company is able to understand its customers buying habits and can identify what promotions they

can do. Through the database management system insights around the customers can be collected, such as:

- Who are the shopaholics users or the customers who shopped more?
- Who are the power users or the customers who paid the most?

Paint Point on Customer Retention: The company would like to understand the customer feedback so they can continuously monitor the customer experience and make sure they come back to buy again. Some insights on the post-buying experience, can be as follows:

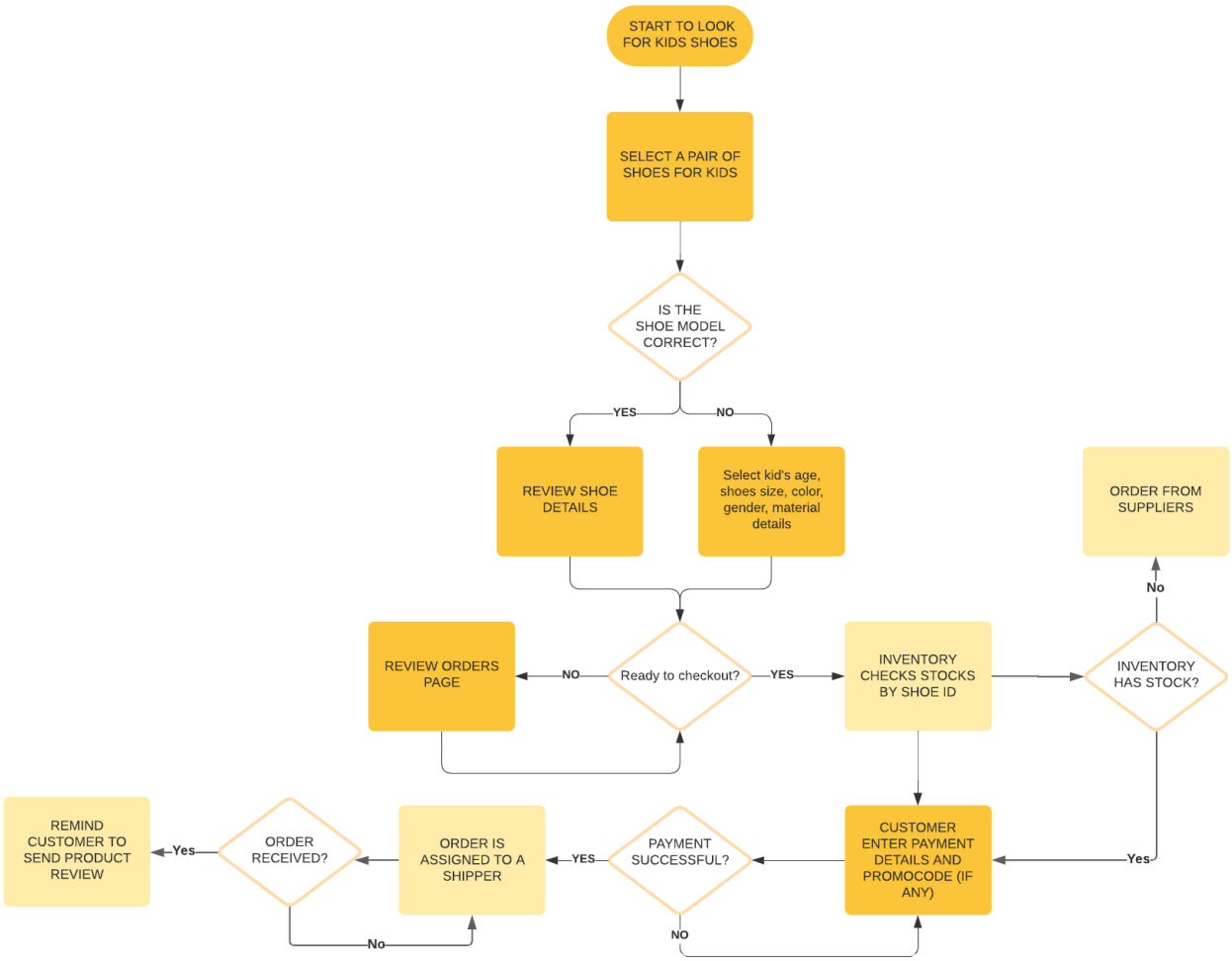
- What is the mean net promoter score out of all the customer reviews for every shoe model?
- If scores 9 to 10 is a promoter, 7 to 8 is a passive, and 0 to 6 is a detractor, how many detractors, passives, and promoters have there ever been?

Pain Point on Operational Costs Savings: Getkicks wants to save on operational costs by proper inventory management. They would like to get insights on:

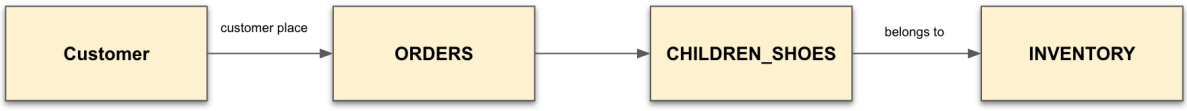
- What is the total price value of a specific type of shoes that are still in stock?
- What shoe products are almost out of stock and who are the suppliers?

Flowchart: Information Flow of System

The flow starts when a custom starts browsing for shoes for his / her kids. There are several shoe models that the customer can choose from. The customer can select a shoe model and selects the kid's age category, shoe size, style, color and a few other qualities and the shop suggests the final shoe product that can be placed as an order. After the payment has been accepted, an order has been placed and it will be assigned to a shipper that delivers the order to the customer.



Key Entities in the System

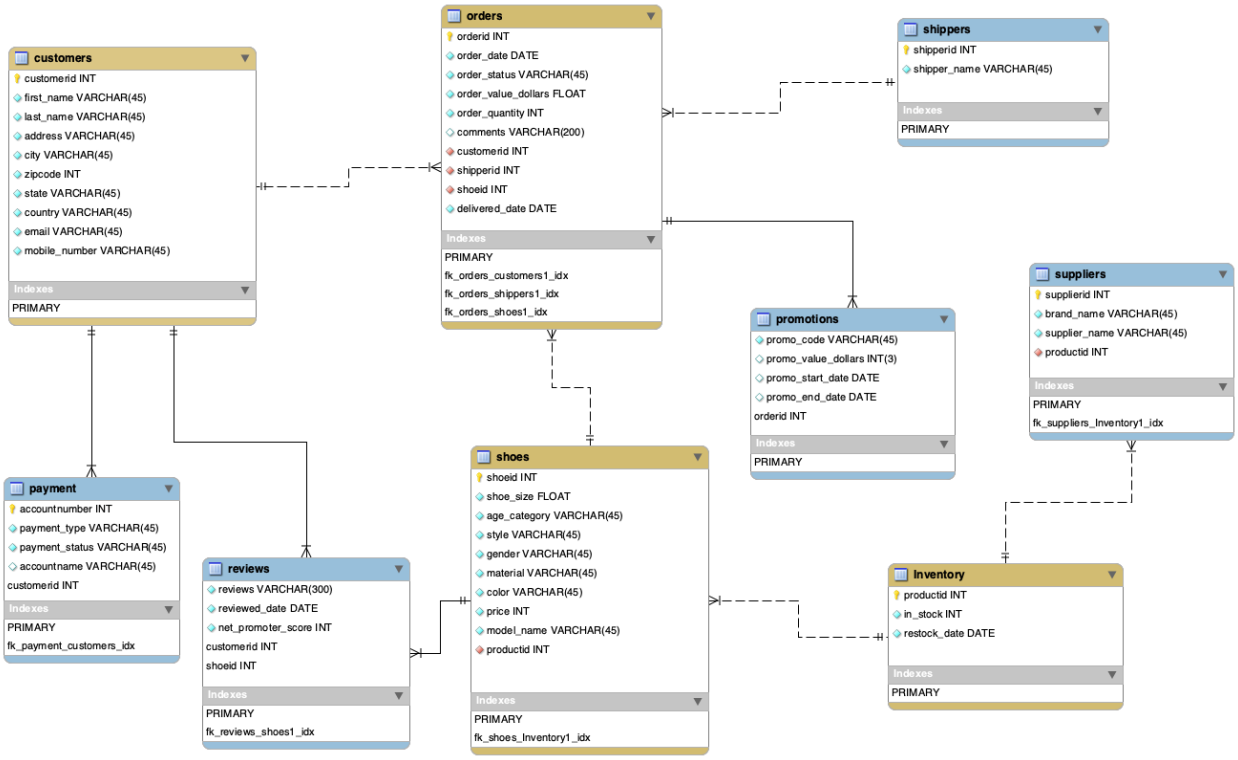


The key entities of the relational database system are the following:

- **Customer:** Collects data about the buyers, reviews as well as the shoes they have ordered
- **Orders:** The orders collects the information on the customer and the shoes bought and connects them to shippers
- **Children_shoes:** This entity or the shoes table have the model and the qualities of the shoes and for what age are the shoes for

- **Inventory:** The inventory is where all the supplies of shoes are managed, it is connected to suppliers who delivers new stocks for the business

Entity-Relational (ER) Model



SQL Queries:

We will be addressing the pain points by

- Who are the top 5 shopaholics users or the customers who shopped more?

```sql
SELECT
c.first_name,
c.last_name,
c.email
FROM customers c, orders o
WHERE c.customerid = o.customerid
ORDER BY o.order_quantity ASC
Limit 5;
```

100% 8:4

Result Grid Filter Rows: Search Export: Fetch rows:

first_name	last_name	email
Kathleen	Lara	aaa@gmail.com
Mia	Hsieh	jjijejn@gmail.com
John	Pearson	akjhkj@gmail.com
Johnny	Lee	kkkkkk@gmail.com
Bianca	Johnson	ccccc2@gmail.com

Result Grid
Form Editor

- Who are the top 5 power users or the customers who paid the most and how much have they spent?

```
SELECT
c.first_name,
c.last_name,
c.email,
SUM(o.order_value_dollars)
FROM customers c, orders o
WHERE c.customerid = o.customerid
GROUP BY
c.first_name,
c.last_name,
c.email,
o.order_value_dollars
Limit 5;
```

100% 9:13

Result Grid | Filter Rows: Search | Export: | Fetch rows:

first_name	last_name	email	SUM(o.order_value_dollar...
Kathleen	Lara	aaa@gmail.com	30
Mia	Hsieh	jjijejn@gmail.com	30
John	Pearson	akjhkj@gmail.com	72
Johnny	Lee	kkkkkk@gmail.com	36
Bianca	Johnson	ccccc2@gmail.com	25

Result Grid | Form Editor

Paint Point on Customer Retention:

- What is the mean net promoter score out of all the customer reviews that the company has for every shoe model?

```
SELECT
s.shoeid,
s.model_name,
avg(r.net_promoter_score)
FROM reviews r, shoes s
WHERE r.shoeid = s.shoeid
GROUP BY
s.shoeid,
s.model_name;
```

00% 14:9

Result Grid | Filter Rows: Search | Export:

shoeid	model_name	avg(r.net_promoter_sco...
1563	baby raptors	10.0000
1564	baby raptors	9.0000
1565	stride rite soft motion genevieve boot	10.0000
1571	baby sneaker beast	4.0000
1566	stride rite soft motion genevieve boot	9.0000
1567	stride rite soft motion genevieve boot	6.0000
1568	stride rite soft motion genevieve boot	10.0000
1569	stride rite soft motion genevieve boot	2.0000
1570	stride rite soft motion genevieve boot	3.0000

- If scores 9 to 10 is a promoter, 7 to 8 is a passive, and 0 to 6 is a detractor, how many detractors, passives, and promoters have there ever been?

```
1  SELECT
2    COUNT(CASE WHEN net_promoter_score >= 9 then 1 ELSE NULL END) as "promoter",
3    COUNT(CASE WHEN net_promoter_score >=7 then 1 ELSE NULL END) as "passive",
4    COUNT(CASE WHEN net_promoter_score >=0 then 1 ELSE NULL END) as "detractor"
5  FROM reviews;
```

100% 14:5

Result Grid | Filter Rows: Search | Export:

promoter	passive	detractor
5	5	9

Result Grid

Pain Point on Operational Costs Savings:

- What is the total price value of a specific type of shoes that are still in stock?

```
1  SELECT
2  s.shoeid,
3  (s.price * i.in_stock) AS total_cost
4  FROM shoes s, inventory i
5  WHERE i.productid = s.productid;
6
```

100% 33:5

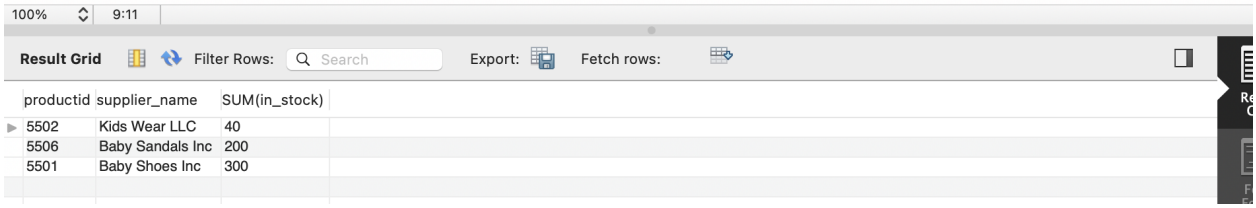
Result Grid | Filter Rows: Search | Export:

shoeid	total_cost
1563	9000
1564	9000
1565	28800
1566	28800
1567	20000
1568	20000
1569	24000
1570	24000
1571	1200
1572	1200
1573	32364
1574	32364
1575	22475
1576	22475
1577	26970
1578	26970

Result Grid
Form Editor
Field Types
Query Stats

- What are the top 3 shoe products that are almost out of stock and who are the suppliers?

```sql
SELECT
i.productid,
s.supplier_name,
SUM(in_stock)
FROM inventory i, suppliers s
WHERE i.productid = s.productid
GROUP BY
i.productid,
s.supplier_name
ORDER BY i.in_stock ASC
Limit 3;
```



100% 9:11

Result Grid | Filter Rows: Search | Export: | Fetch rows:

productid	supplier_name	SUM(in_stock)
5502	Kids Wear LLC	40
5506	Baby Sandals Inc	200
5501	Baby Shoes Inc	300

SQL Procedures

To support different teams from Getkicks who are not very technical, stored procedures were built to make sure that their pain points and objectives are met.

Stored Procedure for Inventory Management Team

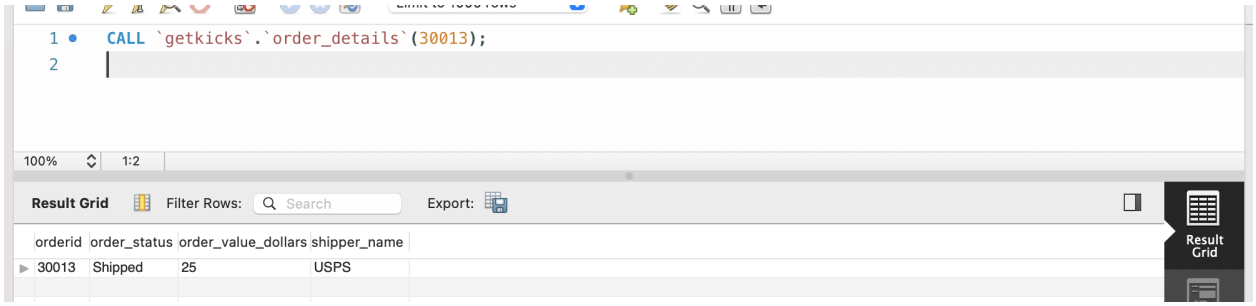
Objective and Key Result: The team would like to quickly pull out order details to easily see the status of specific orders and know how much the company is getting from an order.

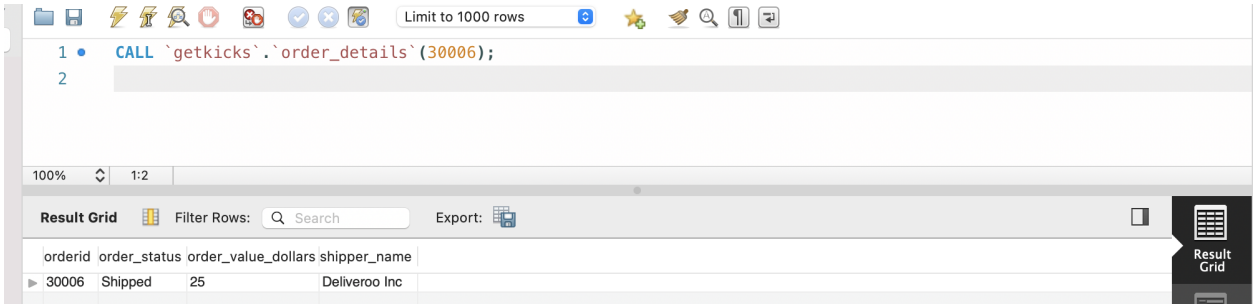
Input / Output Parameters: The team would like to track the status of an order along with its details like orderid, order_status, shipper_name and order_value_dollars. They would like to use the orderid as input to see the records.

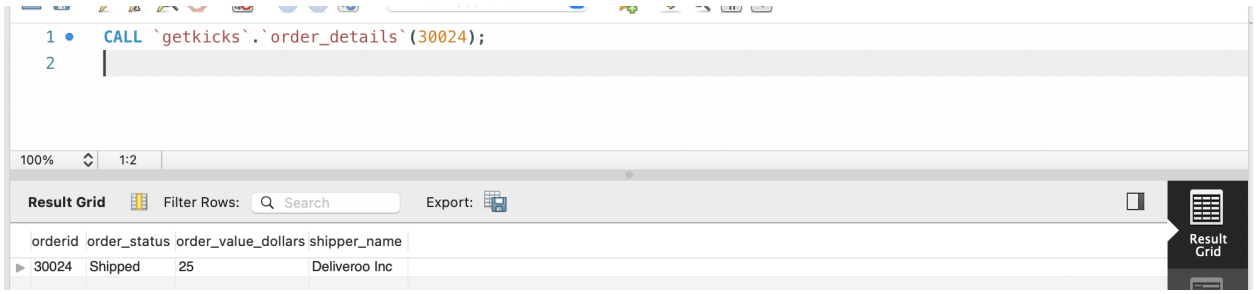
Stored Procedure:

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `order_details`(IN in_order_num INT(5))
BEGIN
SELECT
o.orderid,
o.order_status,
o.order_value_dollars,
s.shipper_name
FROM orders o, shippers s
WHERE o.shipperid = s.shipperid AND
o.orderid = in_order_num;

END
```

Results (3 Examples):







Stored Procedure for Customer Success Team

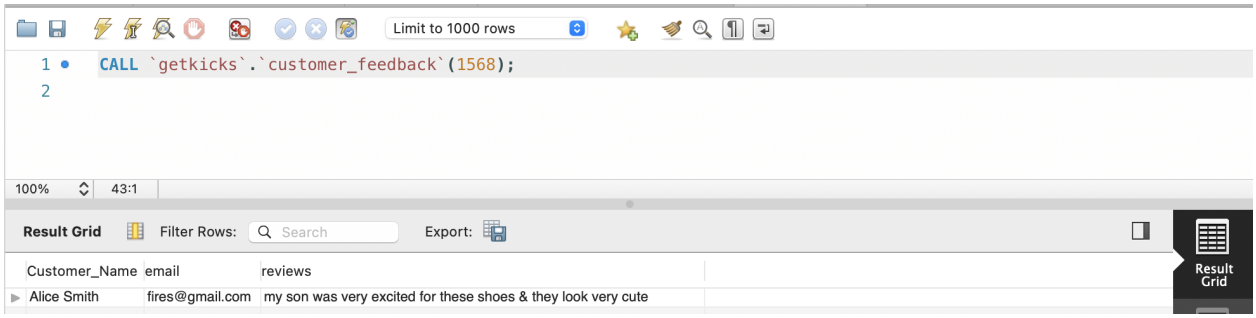
Objective and Key Result: The team would like to track the customer feedback for childrens' shoe products. It is important to see if the customers are getting the correct size for their kids based on the selections they did.

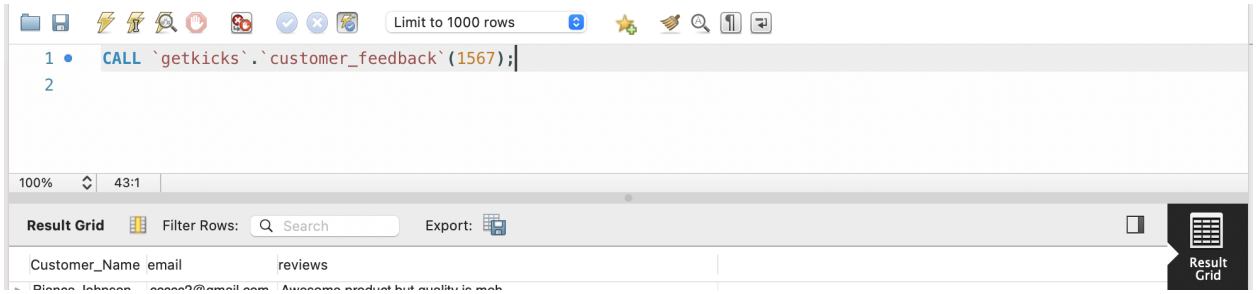
Input / Output Parameters: They would like to use the shoeid as input to see the records and they want to see the reviews, the email of the customer who left the review and the review date.

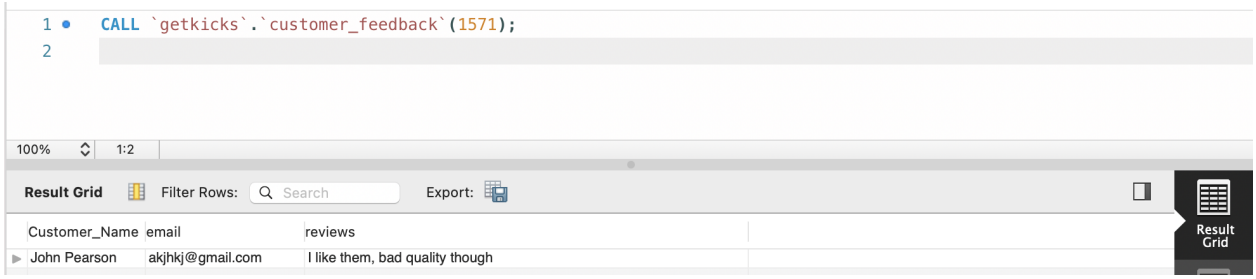
Stored Procedure:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `customer_feedback`(IN in_shoe_id INT(4))
BEGIN
SELECT
concat_ws(" ",c.first_name,c.last_name) AS Customer_Name,
c.email,
r.reviews
FROM customers c, reviews r
WHERE c.customerid=r.customerid AND
r.shoeid = in_shoe_id;
END
```

Results (3 Examples):





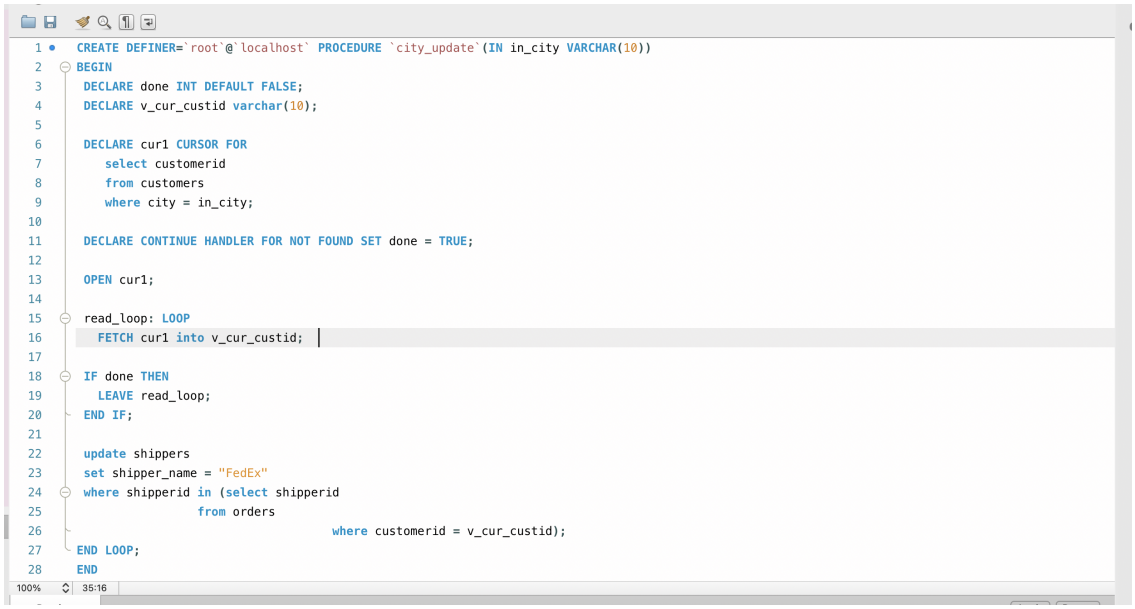


Cursor

Objective and Key Result: Getkicks' goal is to make all the moms/dads ordering from the store happy and satisfied with the service. From ordering up until product delivery. The company noticed that there are several complaints about USPS delivery service in the city of Cambridge which is why it has decided to update the shipper names for those customers living in Cambridge to FedEx.

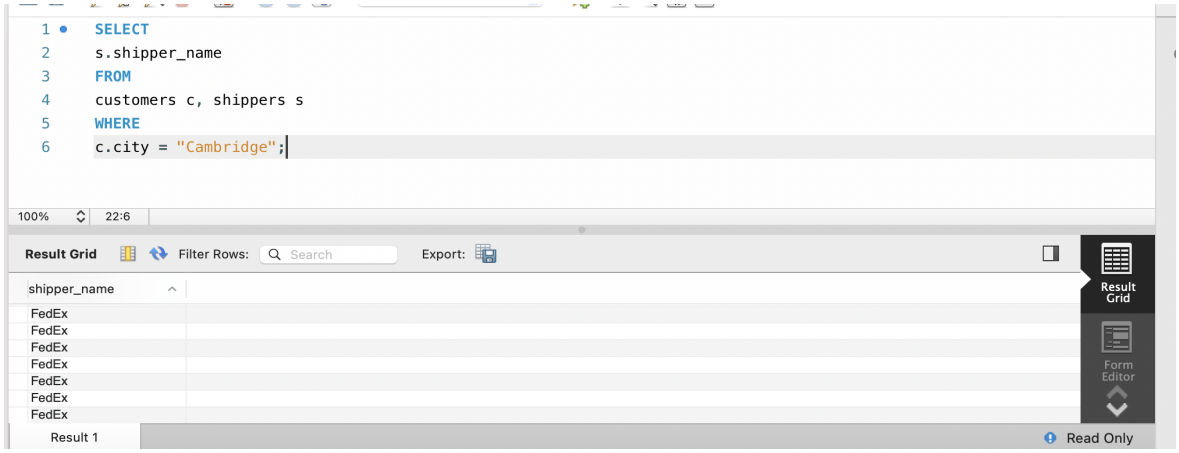
Input / Output Parameters: The procedure cursor will take the city as an input and update the shippers who serve the customers in Cambridge.

Stored Procedure Cursor:



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `city_update`(IN in_city VARCHAR(10))
BEGIN
 DECLARE done INT DEFAULT FALSE;
 DECLARE v_cur_custid varchar(10);

 DECLARE cur1 CURSOR FOR
    select customerid
    from customers
    where city = in_city;

 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

 OPEN cur1;

 read_loop: LOOP
   FETCH cur1 into v_cur_custid;

 IF done THEN
   LEAVE read_loop;
 END IF;

 update shippers
 set shipper_name = "FedEx"
 where shipperid in (select shipperid
                from orders
                                where customerid = v_cur_custid);
 END LOOP;
 END
```

Results (Sample Output to confirm):



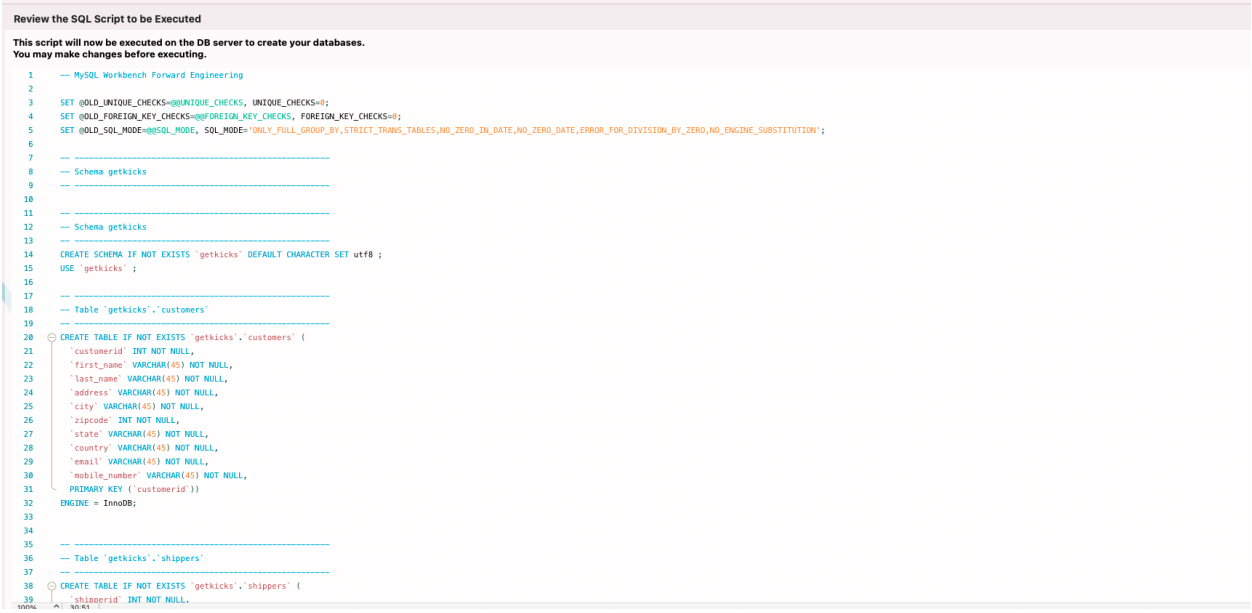
```
SELECT
s.shipper_name
FROM
customers c, shippers s
WHERE
c.city = "Cambridge";
```

Result Grid

shipper_name
FedEx
FedEx
FedEx
FedEx
FedEx
FedEx
FedEx

Appendix

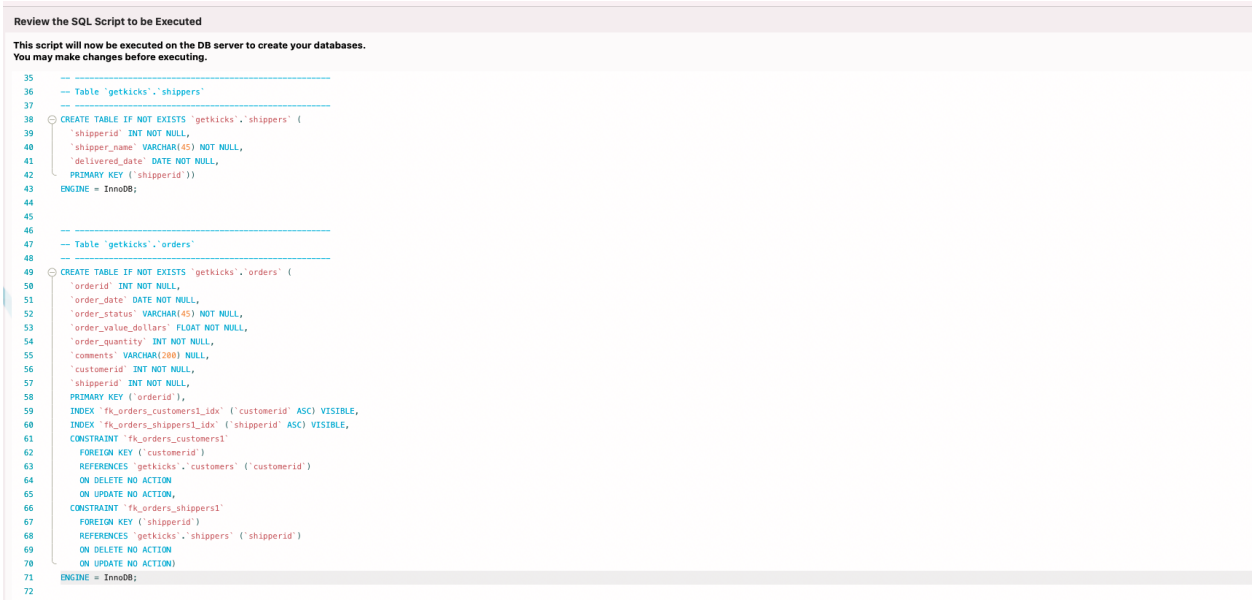
Create Table Scripts: The EER model was forward engineered so a schema and tables can be created.



Review the SQL Script to be Executed

This script will now be executed on the DB server to create your databases.
You may make changes before executing.

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema getkicks
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema getkicks
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `getkicks` DEFAULT CHARACTER SET utf8 ;
USE `getkicks` ;

-- -----------------------------------------------------
-- Table `getkicks`.`customers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`customers` (
  `customerid` INT NOT NULL,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `address` VARCHAR(45) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  `zipcode` INT NOT NULL,
  `state` VARCHAR(45) NOT NULL,
  `country` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `mobile_number` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`customerid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `getkicks`.`shippers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`shippers` (
  `shipperid` INT NOT NULL,
```



Review the SQL Script to be Executed

This script will now be executed on the DB server to create your databases.
You may make changes before executing.

```
-- -----------------------------------------------------
-- Table `getkicks`.`shippers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`shippers` (
  `shipperid` INT NOT NULL,
  `shipper_name` VARCHAR(45) NOT NULL,
  `delivered_date` DATE NOT NULL,
  PRIMARY KEY (`shipperid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `getkicks`.`orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`orders` (
  `orderid` INT NOT NULL,
  `order_date` DATE NOT NULL,
  `order_status` VARCHAR(45) NOT NULL,
  `order_value_dollars` FLOAT NOT NULL,
  `order_quantity` INT NOT NULL,
  `comments` VARCHAR(200) NULL,
  `customerid` INT NOT NULL,
  `shipperid` INT NOT NULL,
  PRIMARY KEY (`orderid`),
  INDEX `fk_orders_customers1_idx` (`customerid` ASC) VISIBLE,
  INDEX `fk_orders_shippers1_idx` (`shipperid` ASC) VISIBLE,
  CONSTRAINT `fk_orders_customers1`
    FOREIGN KEY (`customerid`)
    REFERENCES `getkicks`.`customers` (`customerid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_orders_shippers1`
    FOREIGN KEY (`shipperid`)
    REFERENCES `getkicks`.`shippers` (`shipperid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Review the SQL Script to be Executed

This script will now be executed on the DB server to create your databases.
You may make changes before executing.

```sql
-- -----------------------------------------------------
-- Table `getkicks`.`Inventory`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`Inventory` (
  `productid` INT NOT NULL,
  `in_stock` INT NOT NULL,
  `restock_date` DATE NOT NULL,
  PRIMARY KEY (`productid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `getkicks`.`shoes`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`shoes` (
  `shoeid` INT NOT NULL,
  `shoe_size` FLOAT NOT NULL,
  `age_category` VARCHAR(45) NOT NULL,
  `style` VARCHAR(45) NOT NULL,
  `gender` VARCHAR(45) NOT NULL,
  `material` VARCHAR(45) NOT NULL,
  `color` VARCHAR(45) NOT NULL,
  `price` INT NOT NULL,
  `model_name` VARCHAR(45) NOT NULL,
  `customerid` INT NOT NULL,
  `orderid` INT NOT NULL,
  `productid` INT NOT NULL,
  PRIMARY KEY (`shoeid`),
  INDEX `fk_shoes_customers1_idx` (`customerid` ASC) VISIBLE,
  INDEX `fk_shoes_orders1_idx` (`orderid` ASC) VISIBLE,
  INDEX `fk_shoes_Inventory1_idx` (`productid` ASC) VISIBLE,
  CONSTRAINT `fk_shoes_customers1`
    FOREIGN KEY (`customerid`)
    REFERENCES `getkicks`.`customers` (`customerid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_shoes_orders1`
    FOREIGN KEY (`orderid`)
    REFERENCES `getkicks`.`orders` (`orderid`)
```

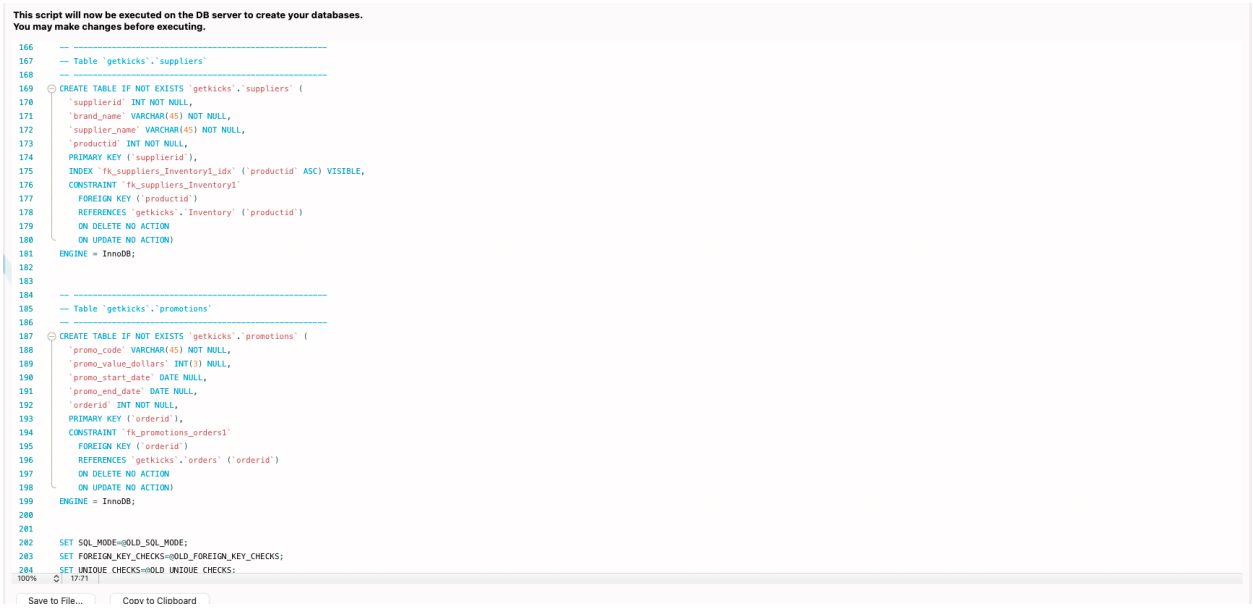
100% 17:71

Review the SQL Script to be Executed

This script will now be executed on the DB server to create your databases.
You may make changes before executing.

```sql
-- -----------------------------------------------------
-- Table `getkicks`.`payment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`payment` (
  `accountnumber` INT NOT NULL,
  `payment_type` VARCHAR(45) NOT NULL,
  `payment_status` VARCHAR(45) NOT NULL,
  `accountname` VARCHAR(45) NULL,
  `customerid` INT NOT NULL,
  PRIMARY KEY (`accountnumber`, `customerid`),
  INDEX `fk_payment_customers_idx` (`customerid` ASC) VISIBLE,
  CONSTRAINT `fk_payment_customers`
    FOREIGN KEY (`customerid`)
    REFERENCES `getkicks`.`customers` (`customerid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `getkicks`.`reviews`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `getkicks`.`reviews` (
  `reviews` VARCHAR(300) NOT NULL,
  `reviewed_date` DATE NOT NULL,
  `net_promoter_score` INT NOT NULL,
  `customerid` INT NOT NULL,
  `shoeid` INT NOT NULL,
  PRIMARY KEY (`customerid`, `shoeid`),
  INDEX `fk_reviews_shoes1_idx` (`shoeid` ASC) VISIBLE,
  CONSTRAINT `fk_reviews_customers1`
    FOREIGN KEY (`customerid`)
    REFERENCES `getkicks`.`customers` (`customerid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_reviews_shoes1`
    FOREIGN KEY (`shoeid`)
    REFERENCES `getkicks`.`shoes` (`shoeid`)
    ON DELETE NO ACTION
```

100% 17:71



References

MySQL & MySQL Workbench: https://www.mysql.com/
Information Flow Chart Diagram: https://lucid.app/documents#/dashboard