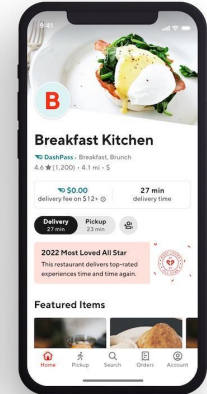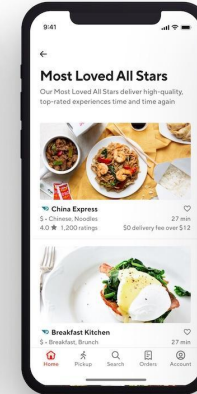DOORDASH

Andrew Lam
Hamun Hodjati
Roy Mao
Hunter Vazirian
Inbar Geva

Company Background

General Description: DoorDash is an online food ordering and delivery service/application that connects customers with local and national businesses.

Their goal is to help local businesses to meet consumers' needs by creating a new way for customers to order and receive food with the company's technology while providing people new ways to earn money by providing delivery jobs accessible to anyone. This infrastructure enables the company to grow and empower local economies.

DoorDash operates in the US, Canada, Australia, Japan, and Germany. Doordash has competitors such as Uber eats and Postmates, but with a 56% market share, DoorDash is the largest food delivery company in the United States. It also has a 60% market share in the convenience delivery category.

User Requirements

1.) DoorDash is a company that has attributes Office ID (Primary Key/Identifier), Office Address (Street, City, Zip), and Employees (Multi-Valued).

2.) Employees have attributes Employee ID (Primary Key/Identifier), Employee Name, Employee Address, and Worker Type. Employees can either be an Office Worker or Delivery Worker, but can not be simultaneously both types. Both Office Workers and Delivery Workers inherit the attributes of Employees, but Office Workers have unique attributes of Annual Salary and Stock Option. Delivery Workers have unique attributes such as License Plate Number and Hourly Wage.

3.) A Customer has attributes of CustomerID (Primary Key/Identifier), Customer Name, Customer Address, and Phone Number. A Customer can place one or many orders, which is a Customer Transaction.

4.) A Customer Transaction has attributes of CustomerOrderID (Primary Key/Identifier), Restaurant, Price, Payment, and Delivery Time. A Customer Transaction is processed by the DoorDash App which has attributes of AppID (Primary Key/Identifier), Search, Orders, and Accounts.

5.) A Delivery Worker uses the DoorDash App to deliver Orders. Every Order has an OrderID (Primary Key/Identifier), Order Date, Order Time, Order Status, and Order Items.

6.) Every Order contains Items. Items are identified with attributes ItemID (Primary Key/Identifier), Total Items, and Order Cost.

7.) Vendors provide Items and are identified with attributes VendorId (Primary Key/Identifier), Vendor Name, Vendor Address (Street, City, Zip)

8.) When a Delivery Worker delivers an Order to a Customer, the customer will receive the Order and the following details will be recorded: Time delivered, Dropped at the Door, or Handed to the Customer options.

9.) After a Customer receives their Order, they can provide feedback. This is recorded as Order Feedback and has attributes of SurveyID (Primary Key/Identifier), Score, Additional Remarks, and Date Recorded

10.) Order Feedback is recorded by the DoorDash App, which is maintained and reviewed by the Office Workers.

Conceptual Data Modeling (E-R Diagram)
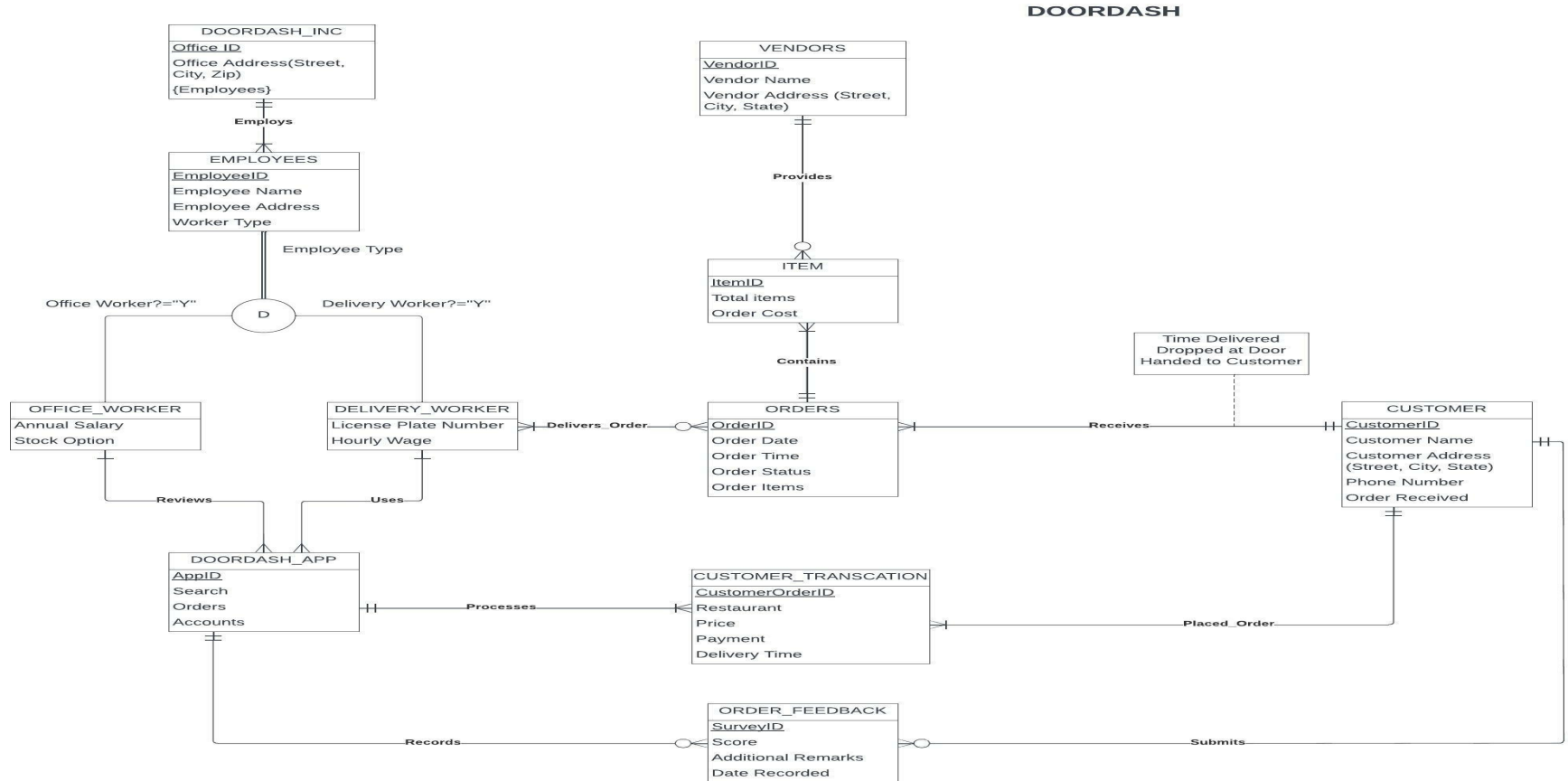
Logical Database Design: Relational Data Model
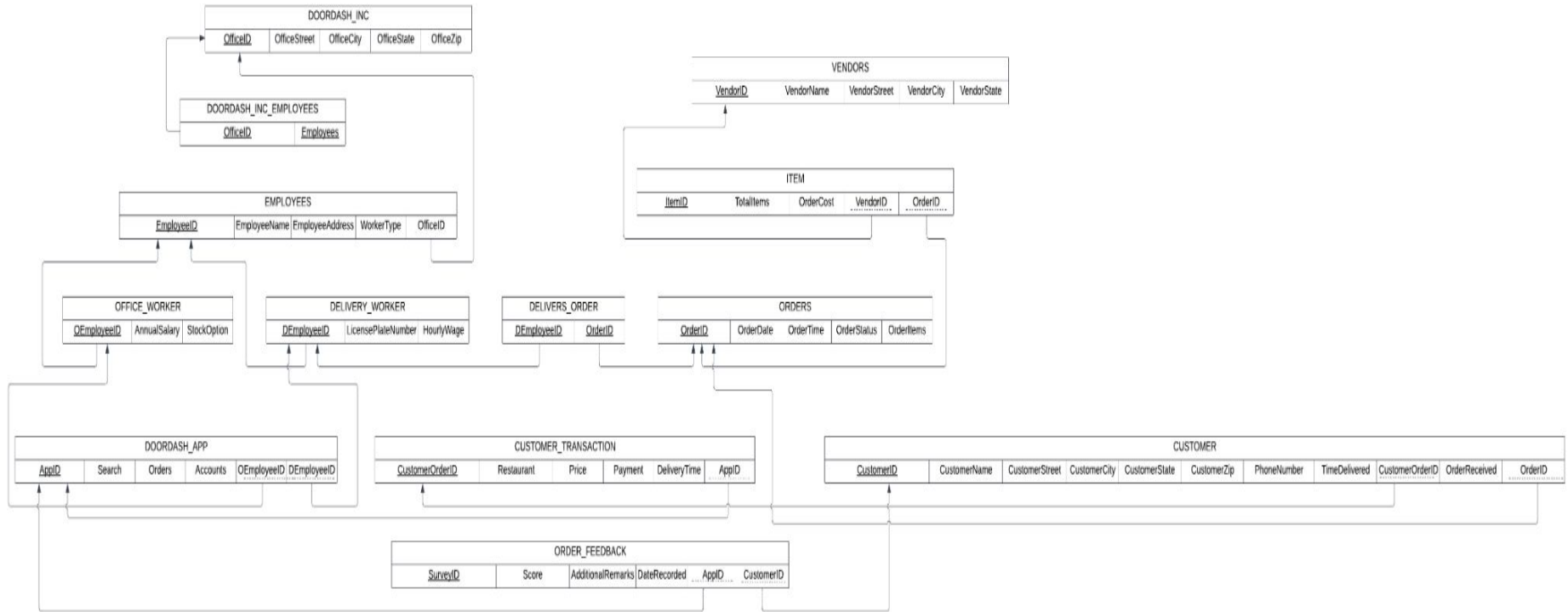
Implementation in MySQL

SQL: Customer Table and Data

```
INSERT INTO customer ( CustomerID, CustomerName, CustomerStreet, CustomerCity,  CustomerState,  CustomerZip,  PhoneNumber,
TimeDelivered,  CustomerOrderID,  OrderReceived, OrderID)
VALUES
(1, 'John Doe', '200 Maple', 'SLC', 'UT', 84102, 45227763, '2009-11-01', 01, 'YES', 1),
(2, 'Bill Doll', '300 West', 'NYC', 'NY', '11424', 45767762,'2022-03-02', '02', 'YES', 2),
(3, 'Josh Turburn', '250 North', 'MIA', 'FL', '47884', 45767733,'2022-03-12', '03', 'NO', 3),
(4, 'Mary Lee', '200 South', 'SLC', 'UT', '84102', 45762763,'2022-03-15', '04', 'YES', 4),
(5, 'Jane Smith', '120 University', 'LA', 'CA', '90815', 45723363, '2022-02-04', '05', 'YES', 5),
(6, 'Luis Smith', '200 Maple', 'HOU', 'TX', '55621', 45127763,'2022-02-10', '06', 'NO', 6),
(7, 'Ben Brown', '300 West', 'CHI', 'IL', '49091', 78762763,'2022-02-04', '07', 'YES', 7),
(8, 'Carl Smith', '250 North', 'SD','CA', '90722', 45768763,'2022-02-11', '08', 'YES', 8),
(9, 'John Doll', '200 South', 'SF', 'CA', '80676', 45127763, '2022-01-11', ' 09', 'YES', 9),
(10, 'Jennet Chris', '120 University', 'SLC', 'UT', '84102', 42227763, '2022-07-05', '010', 'YES', 10);
```

```sql
CREATE TABLE customer (
  CustomerID int(11) NOT NULL,
  CustomerName varchar(30) DEFAULT NULL,
  CustomerStreet varchar(30) DEFAULT NULL,
  CustomerCity varchar(30) DEFAULT NULL,
  CustomerState varchar(30) DEFAULT NULL,
  CustomerZip int(5) DEFAULT NULL,
  PhoneNumber int(20) DEFAULT NULL,
  TimeDelivered date DEFAULT NULL,
  CustomerOrderID int(11) DEFAULT NULL,
  OrderReceived varchar(30) DEFAULT NULL,
  OrderID int(11) DEFAULT NULL,
  CONSTRAINT customer_pk PRIMARY KEY (CustomerID)
)

ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerZip	PhoneNumber	TimeDelivered	CustomerOrderID	OrderReceived	OrderID
1	John Doe	200 Maple	SLC	UT	84102	45227763	2009-11-01	1	YES	1
2	Bill Doll	300 West	NYC	NY	11424	45767762	2022-03-02	2	YES	2
3	Josh Turburn	250 North	MIA	FL	47884	45767733	2022-03-12	3	NO	3
4	Mary Lee	200 South	SLC	UT	84102	45762763	2022-03-15	4	YES	4
5	Jane Smith	120 University	LA	CA	90815	45723363	2022-02-04	5	YES	5
6	Luis Smith	200 Maple	HOU	TX	55621	45127763	2022-02-10	6	NO	6
7	Ben Brown	300 West	CHI	IL	49091	78762763	2022-02-04	7	YES	7
8	Carl Smith	250 North	SD	CA	90722	45768763	2022-02-11	8	YES	8
9	John Doll	200 South	SF	CA	80676	45127763	2022-01-11	9	YES	9
10	Jennet Chris	120 University	SLC	UT	84102	42227763	2022-07-05	10	YES	10

SQL: Employee Table and Data

EmployeeID	EmployeeName	EmployeeAddress	WorkerType	OfficeID
1	James Don	300 Maple	Office	1
2	Bob Cho	400 West	Delivery	2
3	Jonathan Turbo	350 North	Office	3
4	Meredith Li	300 South	Delivery	4
5	Jan Smart	220 University	Office	5
6	Luis Smart	300 Maple	Office	6
7	Bobby Charlton	400 West	Delivery	7
8	Karl Mack	350 North	Delivery	8
9	Jim Beam	300 South	Office	9
10	Janet Jackson	320 University	Office	10

```
1  CREATE TABLE doordash_employees(
2      EmployeeID INT(11) NOT NULL,
3      EmployeeName VARCHAR(30) DEFAULT NULL,
4      EmployeeAddress VARCHAR(30) DEFAULT NULL,
5      WorkerType VARCHAR(30) DEFAULT NULL,
6      OfficeID INT(11) NOT NULL,
7      CONSTRAINT doordash_employees_pk PRIMARY KEY(EmployeeID)
8  ) ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```
INSERT INTO doordash_employees ( EmployeeID, EmployeeName, EmployeeAddress, WorkerType, OfficeID)
VALUES
(1, 'James Don', '300 Maple', 'Office', 1),
(2, 'Bob Cho', '400 West', 'Delivery', 2),
(3, 'Jonathan Turbo', '350 North', 'Office', 3),
(4, 'Meredith Li', '300 South', 'Delivery', 4),
(5, 'Jan Smart', '220 University', 'Office', 5),
(6, 'Luis Smart', '300 Maple', 'Office', 6),
(7, 'Bobby Charlton', '400 West', 'Delivery', 7),
(8, 'Karl Mack', '350 North', 'Delivery', 8),
(9, 'Jim Beam', '300 South', 'Office',  9),
(10, 'Janet Jackson', '320 University', 'Office', 10);
```

SQL: Items Table and Data

```sql
CREATE TABLE item(
    ItemID INT(11) NOT NULL,
    TotalItems INT(10) DEFAULT NULL,
    OrderCost INT(11) DEFAULT NULL,
    VendorID INT(11) DEFAULT NULL,
    OrderID INT(11) DEFAULT NULL,
    CONSTRAINT item_pk PRIMARY KEY(ItemID)
) ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

```sql
INSERT INTO item ( ItemID, TotalItems, OrderCost, VendorID, OrderID)
VALUES
(1, 10, 74, 1, 01),
(2, 9, 33, 2, 02),
(3, 3, 25, 3, 03),
(4, 15, 87, 4, 04),
(5, 22, 120, 5, 05),
(6, 5, 29, 6, 06),
(7, 17, 104, 7, 07),
(8, 10, 120, 8, 08),
(9, 18, 76, 9, 09),
(10, 11, 83, 10, 010);
```

ItemID	TotalItems	OrderCost	VendorID	OrderID
1	10	74	1	1
2	9	33	2	2
3	3	25	3	3
4	15	87	4	4
5	22	120	5	5
6	5	29	6	6
7	17	104	7	7
8	10	120	8	8
9	18	76	9	9
10	11	83	10	10

SQL: Vendors Table and Data

```sql
1  CREATE TABLE vendors(
2      VendorID INT(11) NOT NULL,
3      VendorName VARCHAR(30) DEFAULT NULL,
4      VendorStreet VARCHAR(30) DEFAULT NULL,
5      VendorCity VARCHAR(30) DEFAULT NULL,
6      VendorState VARCHAR(30) DEFAULT NULL,
7      CONSTRAINT vendors_pk PRIMARY KEY(VendorID)
8  ) ENGINE = InnoDB DEFAULT CHARSET = latin1;
```

VendorID	VendorName	VendorStreet	VendorCity	VendorSta
1	McDonalds	1994 Ximeno Avenue	LB	CA
2	KFC	12101 Seal Boulevard	SLC	UT
3	Hole Mole	2290 Bellflower Boulevard	SLC	UT
4	Panda Express	6529 East Spring Street	NYC	NY
5	Little Caesars Pizza	2201 Palo Verde Avenue	LA	CA
6	Jersey Mikes Subs	5726 East 7th Street	HOU	TX
7	Chick-fil-A	4801 2nd Street	CHI	IL
8	Del Taco	5560 East 7th Street	SD	CA
9	Popeyes	3948 East Anaheim Street	SF	CA
10	Chipotle	1230 College Estate	MIA	FL

```sql
INSERT INTO vendors (VendorID, VendorName, VendorStreet, VendorCity, VendorState)
VALUES
(1,'McDonalds', '1994 Ximeno Avenue', 'LB', 'CA'),
(2,'KFC', '12101 Seal Boulevard', 'SLC', 'UT'),
(3,'Hole Mole', '2290 Bellflower Boulevard',  'SLC', 'UT'),
(4,'Panda Express', '6529 East Spring Street', 'NYC', 'NY'),
(5,'Little Caesars Pizza', '2201 Palo Verde Avenue', 'LA', 'CA'),
(6,'Jersey Mikes Subs', '5726 East 7th Street', 'HOU', 'TX'),
(7,'Chick-fil-A', '4801 2nd Street', 'CHI', 'IL'),
(8,'Del Taco', '5560 East 7th Street', 'SD','CA'),
(9,'Popeyes', '3948 East Anaheim Street', 'SF', 'CA'),
(10,'Chipotle', '1230 College Estate',  'MIA', 'FL');
```

Query 1

Customers who live in 'UT' and have received their orders.

```sql
SELECT *
FROM customer
WHERE customer.CustomerState = 'UT'
AND customer.OrderReceived = 'YES';
```

CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerZip	PhoneNumber	TimeDelivered	CustomerOrderID	OrderReceived
1	John Doe	200 Maple	SLC	UT	84102	45227763	2009-11-01	1	YES
4	Mary Lee	200 South	SLC	UT	84102	45762763	2022-03-15	4	YES
10	Jennet Chris	120 University	SLC	UT	84102	42227763	2022-07-05	10	YES

Query 2

Present the unique customer information for all customers who have orders that were delivered with the OrderStatus of 'Handed to Customer'

CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerZip
2	Bill Doll	300 West	NYC	NY	11424
3	Josh Turburn	250 North	MIA	FL	47884
6	Luis Smith	200 Maple	HOU	TX	55621
7	Ben Brown	300 West	CHI	IL	49091

```sql
SELECT DISTINCT
customer.CustomerID, customer.CustomerName, customer.CustomerStreet, customer.CustomerCity ,customer.CustomerState,
customer.CustomerZip
FROM customer
INNER JOIN orders ON
customer.OrderID = orders.OrderID
WHERE orders.OrderStatus = 'Handed to Customer';
```

Query 3

Present the average order cost of items bought by customers for the state of 'CA'

```sql
SELECT AVG(item.OrderCost)
FROM customer, orders, item
WHERE customer.OrderID = orders.OrderID
AND orders.OrderID = item.OrderID
AND customer.CustomerState = 'CA';
```

AVG(item.OrderCost)
105.3333

Query 4

Present the unique itemID, order cost and order cost after 10% discount of all items that exist in orders.

```sql
SELECT DISTINCT item.itemID, item.orderCost, .9*item.orderCost AS '10% Discount'
FROM item NATURAL JOIN orders
```

itemID	orderCost	10% Discount
1	74	66.6
2	33	29.7
3	25	22.5
4	87	78.3
5	120	108.0
6	29	26.1
7	104	93.6
8	120	108.0
9	76	68.4
10	83	74.7

Query 5

Present the total price of each order

```sql
SELECT orders.orderID, SUM(item.totalItems*item.orderCost) AS 'Orders Cost Sum'
FROM orders, item
WHERE orders.orderID = item.orderID
GROUP BY orders.orderID
```

orderID	Orders Cost Sum
1	740
2	297
3	75
4	1305
5	2640
6	145
7	1768
8	1200
9	1368
10	913

First View

Create a VIEW for employees who work in the office

```
1  CREATE VIEW office_employees AS(
2      SELECT
3          doordash_employees.*
4      FROM
5          doordash_employees
6      WHERE
7          doordash_employees.WorkerType LIKE 'o%'
8  );
```

✅ Showing rows 0 - 5 (6 total, Query took 0.0015 seconds.)

SELECT * FROM office_employees

☐ Profiling

☐ Show all | Number of rows: 25 ⌄   Filter rows: Search this table

+ Options

←┬→ ▼	EmployeeID	EmployeeName	EmployeeAddress	WorkerType	OfficeID
☐ 🖉Edit 💠Copy ⊖Delete	1	James Don	300 Maple	Office	1
☐ 🖉Edit 💠Copy ⊖Delete	3	Jonathan Turbo	350 North	Office	3
☐ 🖉Edit 💠Copy ⊖Delete	5	Jan Smart	220 University	Office	5
☐ 🖉Edit 💠Copy ⊖Delete	6	Luis Smart	300 Maple	Office	6
☐ 🖉Edit 💠Copy ⊖Delete	9	Jim Beam	300 South	Office	9
☐ 🖉Edit 💠Copy ⊖Delete	10	Janet Jackson	320 University	Office	10

Second View

Create a VIEW that shows items with an ID over 5 and cost of over 65

```
1  CREATE VIEW itemID5_cost65
2  AS
3   (SELECT item.*
4    FROM item
5    WHERE item.ItemID > 5
6    AND item.OrderCost > 65
7   );
8
```

✅ Showing rows 0 - 3 (4 total, Query took 0.0180 seconds.)

```
SELECT * FROM itemID5_cost65
```

☐ Show all | Number of rows: 25 ⌄   Filter rows: Search this table

+ Options

←T→	⌄	ItemID	TotalItems	OrderCost	VendorID	OrderID
☐ 🖉 Edit ⧉ Copy ⊖ Delete		7	17	104	7	7
☐ 🖉 Edit ⧉ Copy ⊖ Delete		8	10	120	8	8
☐ 🖉 Edit ⧉ Copy ⊖ Delete		9	18	76	9	9
☐ 🖉 Edit ⧉ Copy ⊖ Delete		10	11	83	10	10

First Stored Procedure

Create a stored procedure that lists all employees who have names containing the letter "o"

```
1   DELIMITER //
2
3   CREATE PROCEDURE sp_o_employees ()
4   BEGIN
5   SELECT doordash_employees.EmployeeName
6   FROM doordash_employees
7   WHERE doordash_employees.EmployeeName LIKE '%o%';
8   END //
9
10  DELIMITER ;
```

✔ Showing rows 0 - 4

CALL sp_o_employees

☐ Show all | Num

+ Options

EmployeeName

James Don

Bob Cho

Jonathan Turbo

Bobby Charlton

Janet Jackson

Second Stored Procedure

Using a stored procedure, create a list of all Hole Mole vendors who operate in Salt Lake City

```
1  DELIMITER //
2
3  CREATE PROCEDURE sp_SLC_and_HoleMole_check ()
4  BEGIN
5  SELECT vendors.VendorCity, vendors.VendorName
6  FROM vendors
7  WHERE vendors.VendorCity LIKE '%SLC%' AND vendors.VendorName LIKE '%Hole%';
8  END //
9
10 DELIMITER ;
11
```

✔ Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)

CALL sp_SLC_and_HoleMole_check

☐ Show all | Number of rows: 25 ⌄

+ Options

VendorCity	VendorName
SLC	Hole Mole

Thank You!