```
DROP TABLE IF EXISTS STORE_DEPARTMENT;
DROP TABLE IF EXISTS STORE_PRODUCT;
DROP TABLE IF EXISTS CONTAINS;
DROP TABLE IF EXISTS TRANSACTION;
DROP TABLE IF EXISTS EMPLOYEE;
DROP TABLE IF EXISTS STORE;
DROP TABLE IF EXISTS DEPARTMENT;
DROP TABLE IF EXISTS PRODUCT;
DROP TABLE IF EXISTS CUSTOMER;


CREATE TABLE CUSTOMER
(CUSTOMER_ID INTEGER,
CUSTOMER_FIRST_NAME VARCHAR(30),
CUSTOMER_LAST_NAME VARCHAR(30),
CUSTOMER_STREET_ADDRESS VARCHAR(30),
CUSTOMER_CITY VARCHAR(30),
CUSTOMER_STATE CHAR(2),
CUSTOMER_ZIP_CODE VARCHAR(10),
CUSTOMER_PHONE VARCHAR(14),
CUSTOMER_EMAIL VARCHAR(30),
CONSTRAINT CUSTOMER PRIMARY KEY(CUSTOMER_ID));


CREATE TABLE PRODUCT
(PRODUCT_ID INTEGER,
PRODUCT_TYPE VARCHAR(30),
PRODUCT_PRICE DECIMAL(6,2),
PRODUCT_WEIGHT DECIMAL(6,2),
PRODUCT_QUANTITY INTEGER,
PRODUCT_BRAND VARCHAR(30),
CONSTRAINT PRODUCT PRIMARY KEY(PRODUCT_ID));


CREATE TABLE DEPARTMENT
(DEPARTMENT_ID INTEGER,
DEPARTMENT_TYPE VARCHAR(30),
DEPARTMENT_MANAGER_FIRST_NAME VARCHAR(30),
DEPARTMENT_MANAGER_LAST_NAME VARCHAR(30),
CONSTRAINT DEPARTMENT PRIMARY KEY(DEPARTMENT_ID));
```

```
CREATE TABLE STORE
(STORE_ID INTEGER,
STORE_STREET_ADDRESS VARCHAR(30),
STORE_CITY VARCHAR(30),
STORE_STATE CHAR(2),
STORE_ZIP_CODE VARCHAR(10),
STORE_MANAGER_FIRST_NAME VARCHAR(30),
STORE_MANAGER_LAST_NAME VARCHAR(30),
CONSTRAINT STORE PRIMARY KEY(STORE_ID));

CREATE TABLE EMPLOYEE
(EMPLOYEE_ID INTEGER,
EMPLOYEE_FIRST_NAME VARCHAR(30),
EMPLOYEE_LAST_NAME VARCHAR(30),
EMPLOYEE_STREET_ADDRESS VARCHAR(30),
EMPLOYEE_CITY VARCHAR(30),
EMPLOYEE_STATE CHAR(2),
EMPLOYEE_ZIP_CODE VARCHAR(10),
EMPLOYEE_DATE_EMPLOYED DATE,
STORE_ID INTEGER,
DEPARTMENT_ID INTEGER,
CONSTRAINT EMPLOYEE_PK PRIMARY KEY(EMPLOYEE_ID),
CONSTRAINT EMPLOYEE_STORE_FK FOREIGN KEY(STORE_ID) REFERENCES
STORE(STORE_ID),
CONSTRAINT EMPLOYEE_DEPARTMENT_FK FOREIGN KEY(DEPARTMENT_ID)
REFERENCES DEPARTMENT(DEPARTMENT_ID));


CREATE TABLE TRANSACTION
(TRANSACTION_ID INTEGER,
TRANSACTION_AMOUNT INTEGER,
TRANSACTION_DATE DATE,
CUSTOMER_ID INTEGER,
EMPLOYEE_ID INTEGER,
CONSTRAINT TRANSACTION_PK PRIMARY KEY(TRANSACTION_ID),
CONSTRAINT TRANSACTION_CUSTOMER_FK FOREIGN KEY(CUSTOMER_ID)
REFERENCES CUSTOMER(CUSTOMER_ID),
CONSTRAINT TRANSACTION_EMPLOYEE_FK FOREIGN KEY(EMPLOYEE_ID)
REFERENCES EMPLOYEE(EMPLOYEE_ID));


CREATE TABLE CONTAINS
(TRANSACTION_ID INTEGER,
PRODUCT_ID INTEGER,
ORDERED_QUANTITY INTEGER,
CONSTRAINT CONTAINS_PK PRIMARY KEY (TRANSACTION_ID, PRODUCT_ID),
CONSTRAINT CONTAINS_TRANSACTION_FK FOREIGN KEY (TRANSACTION_ID)
REFERENCES TRANSACTION(TRANSACTION_ID),
CONSTRAINT CONTAINS_PRODUCT_FK FOREIGN KEY(PRODUCT_ID) REFERENCES
PRODUCT(PRODUCT_ID));


CREATE TABLE STORE_PRODUCT
(PRODUCT_ID INTEGER,
STORE_ID INTEGER,
CONSTRAINT STORE_PRODUCT_PK PRIMARY KEY(PRODUCT_ID, STORE_ID),
CONSTRAINT STORE_PRODUCT_PRODUCT_FK FOREIGN KEY(PRODUCT_ID)
REFERENCES PRODUCT(PRODUCT_ID),
CONSTRAINT STORE_PRODUCT_STORE_FK FOREIGN KEY(STORE_ID) REFERENCES
STORE(STORE_ID));


CREATE TABLE STORE_DEPARTMENT
(STORE_ID INTEGER,
DEPARTMENT_ID INTEGER,
CONSTRAINT STORE_DEPARTMENT_PK PRIMARY KEY(STORE_ID, DEPARTMENT_ID),
CONSTRAINT STORE_DEPARTMENT_STORE_FK FOREIGN KEY(STORE_ID)
REFERENCES STORE(STORE_ID),
CONSTRAINT STORE_DEPARTMENT_DEPARTMENT_FK FOREIGN
KEY(DEPARTMENT_ID) REFERENCES DEPARTMENT(DEPARTMENT_ID));



insert into CUSTOMER values
(1, 'Percival', 'Skirrow', '82 Spaight Drive', 'Reston', 'VA', '20195', '703-333-8909',
'pskirrow0@salon.com');
insert into CUSTOMER values
(2, 'Consuelo', 'Gilmartin', '8464 Hooker Parkway', 'New York City', 'NY', '10045', '646-976-5317',
'cgilmartin1@rambler.ru');
insert into CUSTOMER values
(3, 'Boris', 'Greendale', '4 Marcy Park', 'Sacramento', 'CA', '94263', '916-431-3839',
'bgreendale2@fc2.com');
insert into CUSTOMER values
(4, 'Cazzie', 'Witham', '38198 Gina Court', 'Fayetteville', 'NC', '28305', '910-774-0575',
'cwitham3@posterous.com');
insert into CUSTOMER values
(5, 'Traci', 'Greasty', '0 Clemons Lane', 'Rochester', 'NY', '14619', '585-422-8581',
'tgreasty4@google.com.au');
```

```
insert into CUSTOMER values
(6, 'Peri', 'Windram', '95020 North Parkway', 'Evansville', 'IN', '47719', '812-584-8781',
'pwindram5@nsw.gov.au');
insert into CUSTOMER values
(7, 'Charlotta', 'Booley', '1774 Elgar Way', 'Kissimmee', 'FL', '34745', '407-256-5032',
'cbooley6@prweb.com');
insert into CUSTOMER values
(8, 'Malachi', 'Harewood', '92 Lunder Road', 'Lehigh Acres', 'FL', '33972', '239-459-5390',
'mharewood7@multiply.com');
insert into CUSTOMER values
(9, 'Jolie', 'Heinschke', '8 Starling Point', 'Van Nuys', 'CA', '91499', '213-710-1173',
'jheinschke8@livejournal.com');


insert into PRODUCT values
(1,"Produce",7.99,1.20,1,"Cherries");
insert into PRODUCT values
(2,"Produce",2.49,3.23,6,'Organic Fuji Apple');
insert into PRODUCT values
(3,"Produce",1.99,1.02,2,'Organic Garnet Sweet Potato');
insert into PRODUCT values
(4,"Dairy",3.69,1.00,1, 'Shredded Mexican Blend');
insert into PRODUCT values
(5,"Dairy",5.99,1.00,1,'Organic Grade A Milk Whole');
insert into PRODUCT values
(6,"Dairy",5.99,2.00,1,'Total Greek Yogurt');
insert into PRODUCT values
(7,"Meat",15.99,3.09 , 1, "Boneless Beef Ribeye");
insert into PRODUCT values
(8,"Meat",4.99,1.23,1,"Ground Beef 80% Lean/20% Fat");
insert into PRODUCT values
(9,"Meat",5.99,1.00,1,"Unsecured Smokehouse Bacon");


insert into DEPARTMENT values
(1, 'Produce', 'Ellyn', 'Johnson');
insert into DEPARTMENT values
(2, 'Floral', 'Glyn', 'Smith');
insert into DEPARTMENT values
(3, 'Seafood', 'Fabian', 'Garcia');
insert into DEPARTMENT values
(4, 'Deli', 'Lorrin', 'Brown');
insert into DEPARTMENT values
(5, 'Meat', 'Deeanne', 'Miller');
```

```
insert into DEPARTMENT values
(6, 'Dairy', 'Jameson', 'Taylor');
insert into DEPARTMENT values
(7, 'Beer', 'Aubrette', 'Brown');
insert into DEPARTMENT values
(8, 'Bakery', 'Jennifer', 'Davis');
insert into DEPARTMENT values
(9, 'Cheese', 'Allen', 'Williams');


insert into STORE values
(1, '6350 W 3rd Street', 'Los Angeles', 'CA', '90036-0301', 'Larine', 'Walford');
insert into STORE values
(2, '239 N Crescent Dr', 'Beverly Hills', 'CA', '90210-1000', 'Jess', 'Thomson');
insert into STORE values
(3, '1150 Ocean Dr', 'San Francisco', 'CA', '94112-1843', 'Cliff', 'Werner');
insert into STORE values
(4, '100 Pitt St', 'El Paso', 'TX', '79912-1100', 'Lila', 'Morin');
insert into STORE values
(5, '270 Greenwich St', 'New York', 'NY', '10007-1150', 'Leonora', 'Moshi');
insert into STORE values
(6, '316 Kentlands Blvd', 'Gaithersburg', 'MD', '20878-5458', 'Sayre', 'Girodier');
insert into STORE values
(7, '4402 Legendary Dr', 'Destin', 'FL', '32541-0700', 'Nial', 'Furneaux');
insert into STORE values
(8, '4701 N 20th St', 'Phoenix', 'AZ', '85016-4786', 'Alvan', 'Mowat');
insert into STORE values
(9, '2693 Edmondson Rd', 'Cincinnati', 'OH','45209-1910', 'Karen', 'Peck');


insert into EMPLOYEE values
(1, 'Erna', 'Wigfall', '940 Sachs Circle', 'Lake Worth', 'FL', '33467', '2020-03-01', 7, 2);
insert into EMPLOYEE values
(2, 'Gasper', 'Crangle', '51 Iowa Park', 'Fort Wayne', 'IN', '46814', '2020-10-08', 8, 7);
insert into EMPLOYEE values
(3, 'Maureene', 'Phipp', '50 Bunker Hill Place', 'Los Angeles', 'CA', '90071', '2020-02-12', 2, 9);
insert into EMPLOYEE values
(4, 'Farlie', 'Kiossel', '522 Namekagon Plaza', 'Sacramento', 'CA', '95865', '2020-01-31', 8, 7);
insert into EMPLOYEE values
(5, 'Jillayne', 'Skaife d''Ingerthorpe', '56150 Vahlen Place', 'Minneapolis', 'MN', '55470',
'2020-08-29', 5, 1);
insert into EMPLOYEE values
(6, 'Peggie', 'Andrejs', '9050 Reinke Crossing', 'Herndon', 'VA', '22070', '2020-04-11', 8, 5);
insert into EMPLOYEE values
```

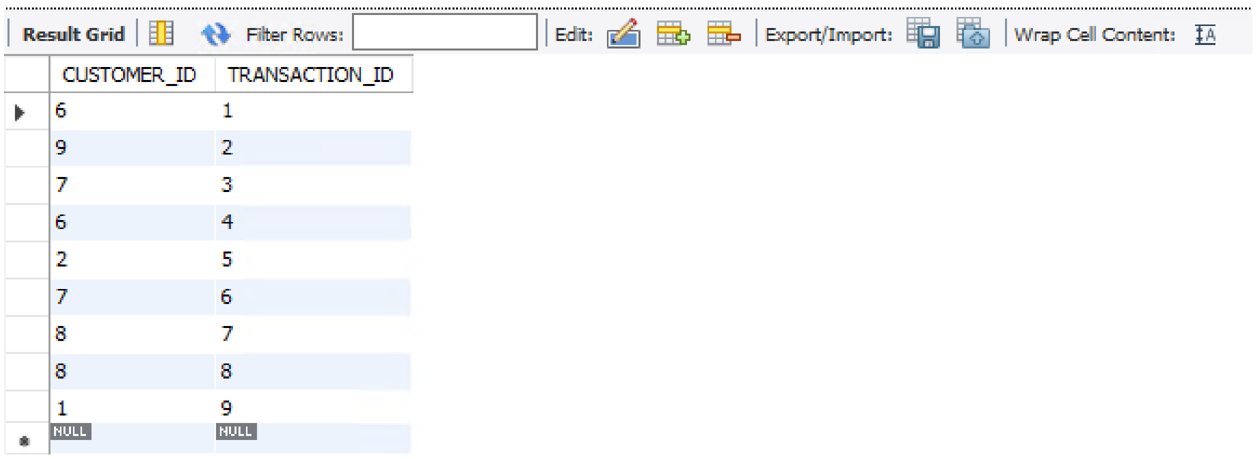
```
(7, 'Sven', 'McShee', '8 Manufacturers Court', 'Sarasota', 'FL', '34276', '2020-08-20', 7, 4);
insert into EMPLOYEE values
(8, 'Skipper', 'Lantry', '2 Scofield Parkway', 'Muskegon', 'MI', '49444', '2020-12-29', 6, 6);
insert into EMPLOYEE values
(9, 'Terese', 'Benger', '1 Reindahl Court', 'Brockton', 'MA', '02305', '2020-02-20', 4, 9);


insert into TRANSACTION values
(1, 12, '2021-03-19', 6, 6);
insert into TRANSACTION values
(2, 20, '2019-11-13', 9, 4);
insert into TRANSACTION values
(3, 19, '2019-04-04', 7, 3);
insert into TRANSACTION
values (4, 78, '2019-05-27', 6, 6);
insert into TRANSACTION values
(5, 32, '2020-10-20', 2, 4);
insert into TRANSACTION values
(6, 26, '2020-10-27', 7, 2);
insert into TRANSACTION values
(7, 43, '2019-01-10', 8, 9);
insert into TRANSACTION
values (8, 64, '2019-08-25', 8, 2);
insert into TRANSACTION values
(9, 77, '2020-06-13', 1, 2);


insert into CONTAINS values
(7, 7, 71);
insert into CONTAINS values
(8, 7, 79);
insert into CONTAINS values
(5, 1, 71);
insert into CONTAINS values
(8, 6, 65);
insert into CONTAINS values
(9, 8, 30);
insert into CONTAINS values
(3, 3, 79);
insert into CONTAINS values
(9, 7, 31);
insert into CONTAINS values
(8, 8, 43);
insert into CONTAINS values
```

```
(1, 6, 17);


insert into STORE_PRODUCT values
(2, 8);
insert into STORE_PRODUCT values
(3, 9);
insert into STORE_PRODUCT values
(4, 7);
insert into STORE_PRODUCT values
(7, 4);
insert into STORE_PRODUCT values
(5, 2);
insert into STORE_PRODUCT values
(7, 3);
insert into STORE_PRODUCT values
(8, 4);
insert into STORE_PRODUCT values
(2, 2);
insert into STORE_PRODUCT values
(1, 4);


insert into STORE_DEPARTMENT values
(8, 1);
insert into STORE_DEPARTMENT values
(3, 4);
insert into STORE_DEPARTMENT values
(1, 1);
insert into STORE_DEPARTMENT values
(1, 8);
insert into STORE_DEPARTMENT values
(3, 7);
insert into STORE_DEPARTMENT values
(4, 8);
insert into STORE_DEPARTMENT values
(9, 6);
insert into STORE_DEPARTMENT values
(4, 5);
insert into STORE_DEPARTMENT values
(3, 9);
```

Part C. Queries

--Query 1--
Purpose: Querying a list of customer transactions that are over $5.00.

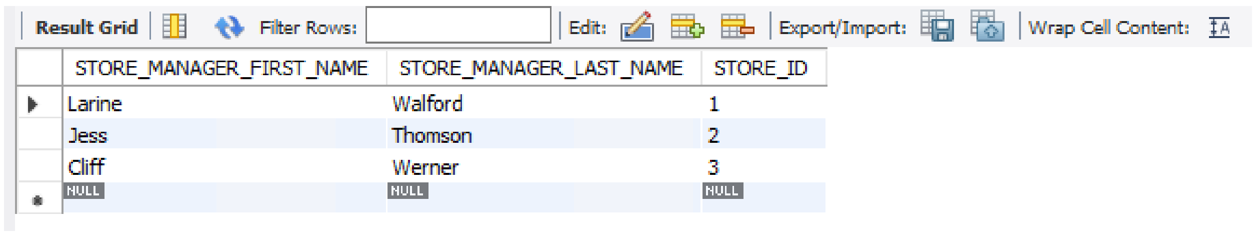
```
SELECT CUSTOMER_ID, TRANSACTION_ID
FROM TRANSACTION
WHERE TRANSACTION_AMOUNT>5.00;
```



CUSTOMER_ID	TRANSACTION_ID
6	1
9	2
7	3
6	4
2	5
7	6
8	7
8	8
1	9
NULL	NULL

--Query 2--
Purpose: Querying a list of store manager's first and last names who work in California.

```
SELECT STORE_MANAGER_FIRST_NAME, STORE_MANAGER_LAST_NAME, STORE_ID
FROM STORE
WHERE STORE_STATE IN ('CA')
```

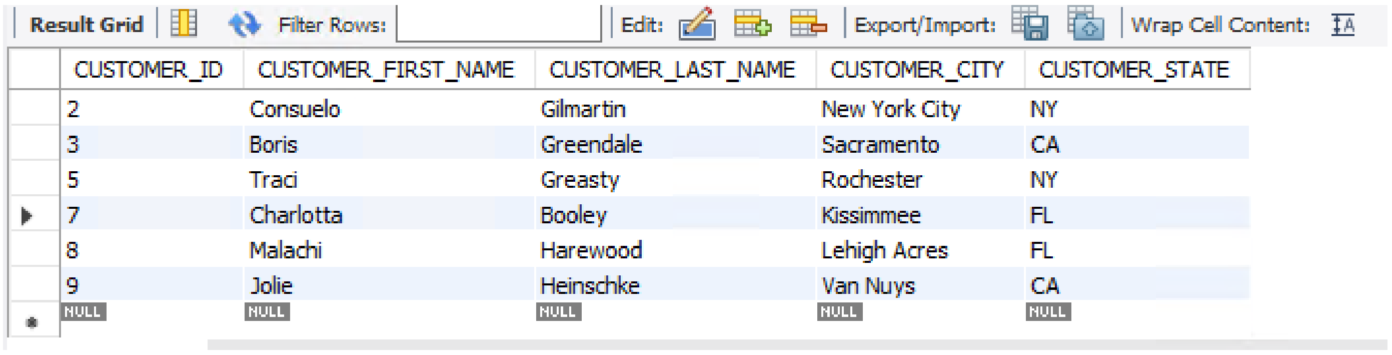


STORE_MANAGER_FIRST_NAME	STORE_MANAGER_LAST_NAME	STORE_ID
Larine	Walford	1
Jess	Thomson	2
Cliff	Werner	3
NULL	NULL	NULL

--Query 3--
Purpose: Querying a sorted list of customers who live in New York, California, and Florida.

```
SELECT CUSTOMER_ID, CUSTOMER_FIRST_NAME, CUSTOMER_LAST_NAME,
CUSTOMER_CITY, CUSTOMER_STATE
```

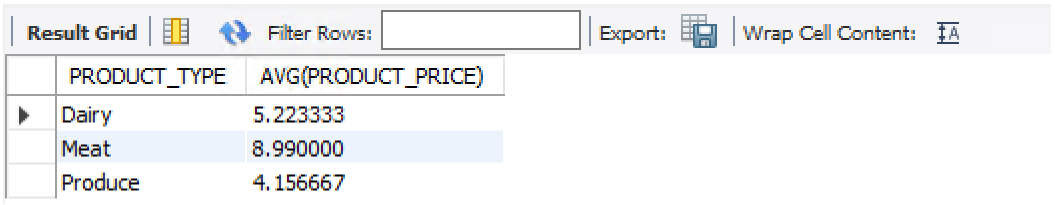
FROM CUSTOMER
WHERE CUSTOMER_STATE IN ('NY', 'CA', 'FL')
ORDER BY CUSTOMER_STATE, CUSTOMER_FIRST_NAME, CUSTOMER_LAST_NAME;



CUSTOMER_ID	CUSTOMER_FIRST_NAME	CUSTOMER_LAST_NAME	CUSTOMER_CITY	CUSTOMER_STATE
2	Consuelo	Gilmartin	New York City	NY
3	Boris	Greendale	Sacramento	CA
5	Traci	Greasty	Rochester	NY
7	Charlotta	Booley	Kissimmee	FL
8	Malachi	Harewood	Lehigh Acres	FL
9	Jolie	Heinschke	Van Nuys	CA
NULL	NULL	NULL	NULL	NULL

--Query 4--
Purpose: Querying a list of the average price (below $9.99) for each product type so that we can identify which product type generates the most income. We chose $9.99 as the threshold for average price because three-digit prices are more attractive to customers than 4 digits (ex.$10.00).

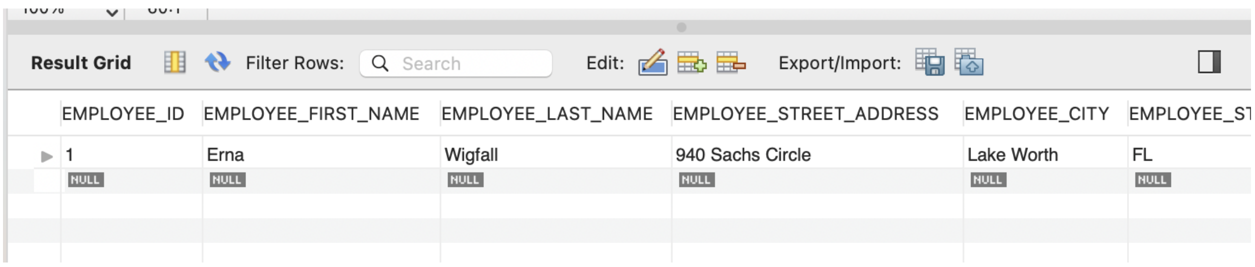
SELECT PRODUCT_TYPE, AVG(PRODUCT_PRICE)
FROM PRODUCT
WHERE PRODUCT_TYPE IN ('PRODUCE', 'MEAT', 'DAIRY')
GROUP BY PRODUCT_TYPE
HAVING AVG(PRODUCT_PRICE)<9.99
ORDER BY PRODUCT_TYPE;



PRODUCT_TYPE	AVG(PRODUCT_PRICE)
Dairy	5.223333
Meat	8.990000
Produce	4.156667

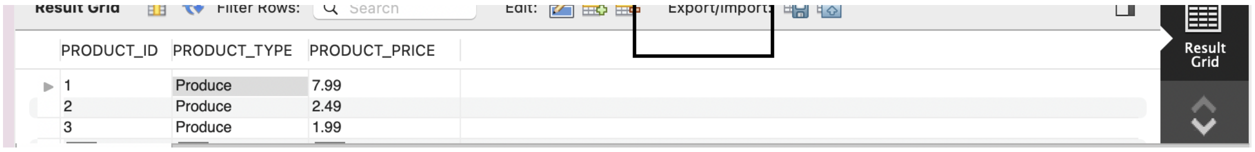
--Query 5--

Purpose: Querying a list of Employee whose first name start with E
SELECT * FROM EMPLOYEE WHERE EMPLOYEE_FIRST_NAME LIKE ‘E%’;



```
EMPLOYEE_ID  EMPLOYEE_FIRST_NAME  EMPLOYEE_LAST_NAME  EMPLOYEE_STREET_ADDRESS  EMPLOYEE_CITY  EMPLOYEE_ST
1            Erna                 Wigfall             940 Sachs Circle         Lake Worth     FL
NULL         NULL                 NULL                NULL                     NULL           NULL
```

--QUERY 6--
PURPOSE: Querying list of Product whose Product type is Produce
SELECT PRODUCT_ID,PRODUCT_TYPE,PRODUCT_PRICE,FROM PRODUCT WHERE PRODUCT_TYPE='Produce';



```
PRODUCT_ID  PRODUCT_TYPE  PRODUCT_PRICE
1           Produce       7.99
2           Produce       2.49
3           Produce       1.99
```

--Query 7--
Shows all employee IDs and their addresses but with a temporary name to make them more readable using an Alias
SELECT EMPLOYEE_ID AS ID, CONCAT(EMPLOYEE_STREET_ADDRESS,', ',EMPLOYEE_CITY,', ',EMPLOYEE_STATE,', ',EMPLOYEE_ZIP_CODE) AS Address
FROM EMPLOYEE;

--Query 8--
Counts the number of employees with addresses in each state that work for whole foods
SELECT EMPLOYEE_STATE, COUNT(EMPLOYEE_STATE)
FROM EMPLOYEE
GROUP BY EMPLOYEE_STATE
ORDER BY EMPLOYEE_STATE;

--Query 9--
Displays the name and address of the customer who made transaction ID 4 using a join query
SELECT CUSTOMER_FIRST_NAME, CUSTOMER_LAST_NAME,
CUSTOMER_STREET_ADDRESS,CUSTOMER_CITY,
CUSTOMER_STATE,CUSTOMER_ZIP_CODE
FROM CUSTOMER, TRANSACTION
WHERE CUSTOMER.CUSTOMER_ID=TRANSACTION.CUSTOMER_ID
AND TRANSACTION_ID=4;

--Query 10--
Shows all the employees who work in the floral department
SELECT DEPARTMENT_TYPE, EMPLOYEE_FIRST_NAME
FROM DEPARTMENT, EMPLOYEE
WHERE

```
DEPARTMENT.DEPARTMENT_ID=EMPLOYEE.DEPARTMENT_ID AND
DEPARTMENT_TYPE='FLORAL';
```

1. For our project, we created a database for the organization Whole Foods Market where we constructed tables, loaded data into them, and created queries for the tables in our database using MySQL. Whole Foods Market is a natural supermarket chain, it has 500 stores in North America and the company employs roughly 95,000 people. For the purposes of this project, we only created 9 tables in total.

   Relationships between entities:
   1. It is an optional many from customer to transaction. A customer can go inside the store but end up buying nothing or many items.
   2. It is a mandatory one from transaction to customer because a transaction can only be made by one and only one customer.
   3. It is a mandatory one from transaction to employee. There has to be an employee who processes the transaction at the checkout counter.
   4. It is an optional many from employee to transaction. Not every employee will be at the checkout counter waiting for a customer to make a transaction such as managers.
   5. It is a one-to-many from transaction to product. A transaction has to contain at least one product.
   6. It is a one-to-many from product to transaction. At least one product must appear in a transaction.
   7. Ordered Quantity relationship attribute is there to know how many of each item is being purchased.
   8. It is a mandatory one-to-many from product to store. Product must show up in at least one store.
   9. It is a mandatory one-to-many from store to product. Store must contain at least one product.
   10. It is a mandatory one-to-many from store to department. Store is assigned to at least one department.
   11.It is an optional many from department to store. Department may or may not show in a store
   12. It is a one-to-many from department to employee. Department is assigned to one or many employees.
   13. It is a mandatory one from employee to department. Employee can work at one and only one department at a time.
   14. It is a mandatory one from employee to store. Employee works for one and only one store at a time.
   15. It is a mandatory one-to-many from store to employee. A store must have at least one employee working there.

3. Tables created for the database and their purpose
   - Customer
     - Purpose: Created to store specific customer attributes such as Customer: ID, First and Last Name, Street Address, City, State, Zip Code, Phone Number, and Email. The specific attribute that will be referenced is Customer ID.
   - Transaction

    - Purpose: Created to store certain transactional attributes such as Transaction ID, Transaction Amount, Transaction Date, Customer ID, and Employee ID. The specific attribute that will be referenced is Transaction ID.
        - // Is Customer ID and Employee ID referenced too?
- Employee
    - Purpose: Created to store specific attributes that relate to one Employee entity. Attributes include Employee: ID, First and Last Name, Street Address, City, State, Zip Code, Date Employed, their Store ID, and Department ID. The specific attribute that will be referenced is Employee ID.
    - // Is Store ID and Department ID referenced too?
- Store
    - Purpose: Created to store varying store attributes that correspond to one store entity. Attributes include Store: ID, Street Address, City, State, Zip Code, Manager First Name, and Manager Last Name. The specific attribute that will be referenced is Store ID.
- Department
    - Purpose: Created to store specific attributes corresponding to a Department entity. Attributes include Department: ID, Type, Manager First Name, and Manager Last Name. The specific attribute that will be referenced is Department ID.
- Product
    - Purpose: Created to store specific attributes that correspond to a Product. Attributes include Product: ID, Type, Price, Weight, Quantity, and Brand. The specific attribute that will be referenced is the Product ID.
- Contains
    - Purpose: Created to predict which entities contain specific attributes such as Transaction ID, Product ID, and Quantity. The specific attributes that will be referenced are Transaction and Product ID.
- Store Product
    - Purpose: Created to predict which Store contains a specific product. The specific attributes that will be referenced are Product and Store ID.
- Store Department
    - Purpose: Created to predict which Store contains a specific department. The specific attributes that will be referenced are Store ID and Department ID.

4. Queries created from the database
- Query 1
    - Purpose: Querying a list of customer transactions that are over $5.00.
- Query 2
    - Purpose: Querying a list of store manager's first and last names who work in California.
- Query 3
    - Purpose: Querying a sorted list of customers who live in New York, California, and Florida.

- Query 4
  - Purpose: Querying a list of the average price (below $9.99) for each product type so that we can identify which product type generates the most income. We chose $9.99 as the threshold for average price because three-digit prices are more attractive to customers than 4 digits (ex.$10.00).
- Query 5
  - Purpose: Querying a list of Employees whose first name starts with E.
- Query 6
  - Purpose: Querying list of Products whose Product type is Produce.
- Query 7
  - Purpose: Shows all employee IDs and their addresses but with a temporary name to make them more readable.
- Query 8
  - Purpose: Count the number of employees with addresses in each state that work for whole foods.
- Query 9
  - Purpose: Displays the name and address of the customer who made transaction ID 4.
- Query 10
  - Purpose: Shows all the employees who work in the floral department.