A2_Liu_Eliza

1. Import the Fortune500_2019 dataset using the Oracle cloud SQL developer interface data loading option. Compare commands for selecting columns using R, Python, and SQL for the Fortune500_2019 dataset. Provide screenshots of results. **(7 points)**

SQL: SELECT * FROM Fortune500_2019;
SELECT Company, Revenue, Profits

FROM Fortune500_2019;

SELECT Company, Revenue, Profits

FROM Fortune500_2019

WHERE Revenue > 50000;

SELECT Company, Revenue, PROFITS

FROM Fortune500_2019

ORDER BY Revenue DESC;

```
1  SELECT * FROM Fortune500_2019;
2  SELECT Company, Revenue, Profits
3  FROM Fortune500_2019;
4  SELECT Company, Revenue, Profits
5  FROM Fortune500_2019
6  WHERE Revenue > 50000;
7  SELECT Company, Revenue, PROFITS
8  FROM Fortune500_2019
9  ORDER BY Revenue DESC;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace

Download ▾ Execution time: 0.012 seconds

	COMPANY	REVENUE	PROFITS
1	Walmart	514405	6670
2	Exxon Mobil	290212	20840
3	Apple	265595	59531
4	Berkshire Hathaway	247837	4021
5	Amazon.com	232887	10073
6	UnitedHealth Group	226247	11986

R:
Fortune500_2019<- read.csv("Downloads/BSAN 726/Week_3 DataManagement/Monday Class/Fortune500_2019.csv")

selected_columns <- Fortune500_2019 %>%
select(Company, Revenue, Profits)
print(selected_columns)

```{r}
Fortune500_2019<- read.csv("Downloads/BSAN 726/Week_3 DataManagement/Monday Class/Fortune500_2019.csv")
selected_columns <- Fortune500_2019 %>%
select(Company, Revenue, Profits)
print(selected_columns)
```

Description: df [500 × 3]

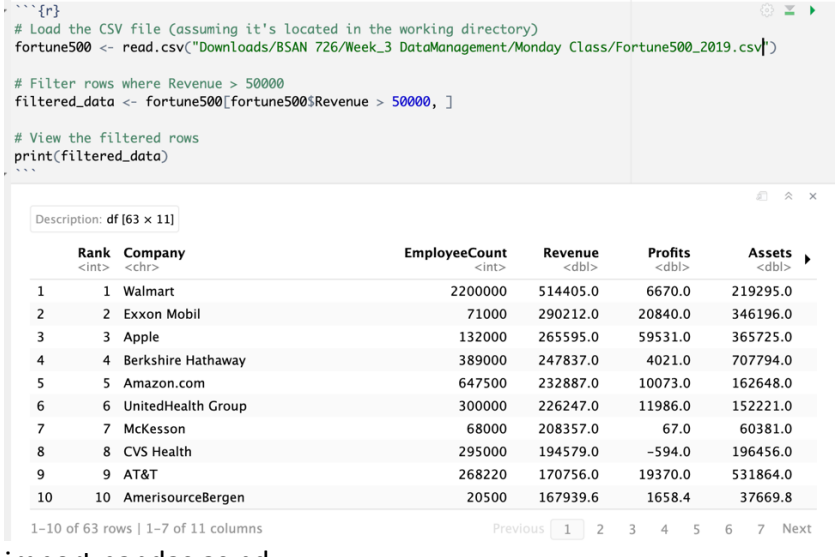
Company <chr>	Revenue <dbl>	Profits <dbl>
Walmart	514405.0	6670.0
Exxon Mobil	290212.0	20840.0
Apple	265595.0	59531.0
Berkshire Hathaway	247837.0	4021.0
Amazon.com	232887.0	10073.0
UnitedHealth Group	226247.0	11986.0
McKesson	208357.0	67.0
CVS Health	194579.0	-594.0
AT&T	170756.0	19370.0
AmerisourceBergen	167939.6	1658.4

Python:
import pandas as pd
fortune500 = pd.read_csv ("Fortune500_2019.csv")
selected_columns = fortune500[['Company', 'Revenue', 'Profits']]
print(selected_columns)

```python
import pandas as pd

# Load the CSV
fortune500 = pd.read_csv("Fortune500_2019.csv")

# Select specific columns
selected_columns = fortune500[['Company', 'Revenue', 'Profits']]

print(selected_columns)
```

```
                  Company   Revenue  Profits
0                 Walmart  514405.0   6670.0
1             Exxon Mobil  290212.0  20840.0
2                   Apple  265595.0  59531.0
3       Berkshire Hathaway  247837.0   4021.0
4              Amazon.com  232887.0  10073.0
..                    ...       ...      ...
495  Simon Property Group    5657.9   2440.1
496              Navient    5610.0    395.0
497        Western Union    5589.9    851.9
498       Peabody Energy    5581.8    646.9
499         Levi Strauss    5575.4    283.1

[500 rows x 3 columns]
```

2. In SQL, we use the *where* clause to selectively filter rows. How can we filter rows in R and Python? Give simple examples using the Fortune500_2019 dataset. Provide screenshots of results. **(8 points)**
   # Load the CSV file (assuming it's located in the working directory)
   fortune500 <- read.csv("Fortune500_2019.csv")
   # Filter rows where Revenue > 50000
   filtered_data <- fortune500[fortune500$Revenue > 50000,]
   # View the filtered rows
   print(filtered_data)



```{r}
# Load the CSV file (assuming it's located in the working directory)
fortune500 <- read.csv("Downloads/BSAN 726/Week_3 DataManagement/Monday Class/Fortune500_2019.csv")

# Filter rows where Revenue > 50000
filtered_data <- fortune500[fortune500$Revenue > 50000, ]

# View the filtered rows
print(filtered_data)
```

Description: df [63 × 11]

	Rank <int>	Company <chr>	EmployeeCount <int>	Revenue <dbl>	Profits <dbl>	Assets <dbl>
1	1	Walmart	2200000	514405.0	6670.0	219295.0
2	2	Exxon Mobil	71000	290212.0	20840.0	346196.0
3	3	Apple	132000	265595.0	59531.0	365725.0
4	4	Berkshire Hathaway	389000	247837.0	4021.0	707794.0
5	5	Amazon.com	647500	232887.0	10073.0	162648.0
6	6	UnitedHealth Group	300000	226247.0	11986.0	152221.0
7	7	McKesson	68000	208357.0	67.0	60381.0
8	8	CVS Health	295000	194579.0	-594.0	196456.0
9	9	AT&T	268220	170756.0	19370.0	531864.0
10	10	AmerisourceBergen	20500	167939.6	1658.4	37669.8

1–10 of 63 rows | 1–7 of 11 columns Previous 1 2 3 4 5 6 7 Next

```
import pandas as pd

# Load the CSV file

fortune500 = pd.read_csv("Fortune500_2019.csv")

# Filter rows where Revenue > 50000

filtered_data = fortune500[fortune500['Revenue'] > 50000]

# View the filtered rows
```

print(filtered_data)

Captured with Xnip

```python
import pandas as pd

# Load the CSV file
fortune500 = pd.read_csv("Fortune500_2019.csv")

# Filter rows where Revenue > 50000
filtered_data = fortune500[fortune500['Revenue'] > 50000]

# View the filtered rows
print(filtered_data)
```

```
    Rank                Company  EmployeeCount   Revenue  Profits    Assets  \
0      1                Walmart        2200000  514405.0   6670.0  219295.0
1      2            Exxon Mobil          71000  290212.0  20840.0  346196.0
2      3                  Apple         132000  265595.0  59531.0  365725.0
3      4      Berkshire Hathaway         389000  247837.0   4021.0  707794.0
4      5              Amazon.com         647500  232887.0  10073.0  162648.0
..   ...                     ...            ...       ...      ...       ...
58    59  Energy Transfer Equity          11768   54436.0   1694.0   88246.0
59    60         Lockheed Martin         105000   53762.0   5046.0   44876.0
60    61                  Pfizer          92400   53647.0  11153.0  159422.0
61    62     Goldman Sachs Group          36600   52528.0  10459.0  931796.0
62    63          Morgan Stanley          60348   50193.0   8748.0  853531.0

    MarketValue              Sector  \
0      279880.3           Retailing
1      342172.0              Energy
2      895667.4          Technology
3      493870.3          Financials
4      874709.5          Technology
..          ...                 ...
58      40260.0              Energy
59      84887.6  Aerospace & Defense
60     235785.1         Health Care
61      70414.9          Financials
62      72110.8          Financials

                                Industry       Hqcity Hqstate
0                   General Merchandisers  Bentonville      AR
1                      Petroleum Refining       Irving      TX
2                Computers, Office Equipment    Cupertino      CA
3   Insurance: Property and Casualty (Stock)        Omaha      NE
4            Internet Services and Retailing      Seattle      WA
..                                       ...          ...     ...
58                             Pipelines       Dallas      TX
59                   Aerospace and Defense     Bethesda      MD
60                        Pharmaceuticals     New York      NY
61                      Commercial Banks     New York      NY
62                      Commercial Banks     New York      NY

[63 rows x 11 columns]
```

3.

```
DROP TABLE PUR_CUSTOMER ;
DROP TABLE PUR_PRODUCT ;
DROP TABLE PUR_SALES ;
DROP TABLE PUR_SALESPERSON ;
DROP TABLE PUR_SALES_CONTACT ;

CREATE TABLE PUR_CUSTOMER (
    CUSTOMER_ID INTEGER PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    LAST_NAME VARCHAR2(50),
    ADDRESS VARCHAR2(80),
    CITY VARCHAR2(40),
    COUNTRY VARCHAR2(50),
    DATE_ADDED DATE,
    REGION VARCHAR2(10),
    EMAIL VARCHAR2(100) UNIQUE
) ;
CREATE TABLE PUR_PRODUCT (
    PRODUCT_ID INTEGER PRIMARY KEY,
    PRODUCT_NAME VARCHAR2(100),
    STANDARD_COST NUMBER(10,2),
    LIST_PRICE NUMBER(10,2)
) ;
CREATE TABLE PUR_SALES (
    SALES_ID INTEGER PRIMARY KEY,
    SALES_DATE DATE,
    ORDER_ID INTEGER,
    PRODUCT_ID INTEGER,
    CUSTOMER_ID INTEGER,
    UNIT_PRICE NUMBER(10,2),
    DISCOUNT NUMBER(10,2),
    TOTAL_AMOUNT NUMBER(10,2),
    FOREIGN KEY (PRODUCT_ID) REFERENCES PUR_PRODUCT(PRODUCT_ID),
    FOREIGN KEY (CUSTOMER_ID) REFERENCES PUR_CUSTOMER(CUSTOMER_ID)
);
CREATE TABLE PUR_SALESPERSON (
    SALESPERSON_ID INTEGER PRIMARY KEY,
    JOB_TITLE VARCHAR2(80),
    FIRST_NAME VARCHAR2(50),
    LAST_NAME VARCHAR2(50),
    MANAGER VARCHAR2(20)
);
CREATE TABLE PUR_SALES_CONTACT (
    SALES_ID INTEGER,
    SALESPERSON_ID INTEGER,
    COMMISSION NUMBER(6,2),
```

```
    PRIMARY KEY (SALES_ID, SALESPERSON_ID),
    FOREIGN KEY (SALES_ID) REFERENCES PUR_SALES(SALES_ID),
    FOREIGN KEY (SALESPERSON_ID) REFERENCES PUR_SALESPERSON(SALESPERSON_ID)
);
```

4.

	ıER_ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	COUNTRY	DATE_ADDED	REGION	EMAIL
1	16	JAMES	SMITH	555 Willie Stargell A	Alameda	USA	2/24/2021, 12:00:00	WEST	james.s@cooln
2	10	JOHN	DOE	200 Clinton Pkwy	Lawrence	USA	1/12/2021, 12:00:00	MIDWEST	johnny.d@cool
3	15	DIANA	THOMAS	1010 East Roger St	Miami	USA	1/12/2021, 12:00:00	SOUTH	diana.t@bright
4	11	JANE	DOE	201 Clinton Pkwy	Lawrence	USA	2/12/2021, 12:00:00	MIDWEST	jane.d@fancym
5	12	ALICA	EDWARD	3456 Gates Dr	Chicago	USA	2/12/2021, 12:00:00	MIDWEST	alica.e@funmai
6	13	PETER	EDWARD	123 Sandy Dr	Phoenix	USA	3/2/2021, 12:00:00 A	WEST	peter.e@happy
7	14	DAVE	TAYLOR	2345 Petersburg St	Austin	USA	3/2/2021, 12:00:00 A	WEST	dave.t@joymail
8	17	SARAH	JOHNSON	190 E. Stacy Rd	Allen	USA	5/7/2021, 12:00:00 A	SOUTH	sarah.j@fancyr
9	18	EMILY	WILSON	1168 State College B	Anaheim	USA	5/12/2021, 12:00:00	WEST	emily.w@funma

	PRODUCT_ID	PRODUCT_NAME	STANDARD_COST	LIST_PRICE
1	500	Airpods Pro	160	200
2	100	Mobile Cover	30	35
3	300	LG F100	100	15
4	400	Apple A100	110	125
5	200	Samsung F7100	80	100
6	600	MacBook	1000	1200
7	700	Sony A9	200	225

	SALES_ID	SALES_DATE	ORDER_ID	PRODUCT_ID	CUSTOMER_ID	UNIT_PRICE	DISCOUNT	TOTAL_AMOUNT
1	1005	10/16/2021, 12:00:0(	400	400	14	50	*(null)*	125
2	1009	12/24/2021, 12:00:0(	700	700	15	110	*(null)*	150
3	1004	11/25/2021, 12:00:0(	300	300	13	40	*(null)*	80
4	1006	11/20/2021, 12:00:0(	500	500	12	75	*(null)*	100
5	1007	7/2/2021, 12:00:00 A	600	600	15	20	*(null)*	50
6	1008	9/30/2021, 12:00:00	500	500	17	100	*(null)*	200
7	1010	11/11/2021, 12:00:00	400	400	18	50	*(null)*	75
8	1001	8/12/2021, 12:00:00	100	100	11	40	*(null)*	440
9	1002	9/22/2021, 12:00:00	200	200	13	35	*(null)*	50
10	1003	8/25/2021, 12:00:00	300	300	13	40	*(null)*	80

	SALESPERSON_ID	JOB_TITLE	FIRST_NAME	LAST_NAME	MANAGER
1	11001	Developer	Anita	Borg	Greg
2	11002	Customer Facing	Samantha	Doe	John
3	11004	Entry Level	Tom	Pardi	Jane
4	11006	Customer Facing	Olivia	White	John
5	11000	Entry Level	Ben	Thomas	Greg
6	11003	Entry Level	Sue	Bellman	Jane
7	11005	Sales Consultant	David	Brown	John
8	11007	Developer	Robert	Taylor	Greg

	SALES_ID	SALESPERSON_ID	COMMISSION
1	1001	11003	4.3
2	1004	11004	10.5
3	1001	11004	4.5
4	1005	11006	2.5
5	1007	11007	5.5
6	1010	11007	11.1
7	1002	11001	30.92
8	1003	11002	9.9
9	1006	11005	12.6
10	1008	11005	20.2
11	1009	11006	40.9

5.1

```
-- Update the city for customer Dave Taylor with CUSTOMER_ID 14
UPDATE PUR_CUSTOMER
SET CITY = 'Austin'
WHERE CUSTOMER_ID = 14;
-- Display the updated PUR_CUSTOMER table
SELECT * FROM PUR_CUSTOMER;
```

	IER_ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	COUNTRY	DATE_ADDED	REGION	EMAIL
1	16	JAMES	SMITH	555 Willie Stargell A	Alameda	USA	2/24/2021, 12:00:00	WEST	james.s@coolm
2	10	JOHN	DOE	200 Clinton Pkwy	Lawrence	USA	1/12/2021, 12:00:00	MIDWEST	johnny.d@coolr
3	15	DIANA	THOMAS	1010 East Roger St	Miami	USA	1/12/2021, 12:00:00	SOUTH	diana.t@brightr
4	11	JANE	DOE	201 Clinton Pkwy	Lawrence	USA	2/12/2021, 12:00:00	MIDWEST	jane.d@fancym
5	12	ALICA	EDWARD	3456 Gates Dr	Chicago	USA	2/12/2021, 12:00:00	MIDWEST	alica.e@funmai
6	13	PETER	EDWARD	123 Sandy Dr	Phoenix	USA	3/2/2021, 12:00:00 A	WEST	peter.e@happy
7	14	DAVE	TAYLOR	2345 Petersburg St	Austin	USA	3/2/2021, 12:00:00 A	WEST	dave.t@joymail.
8	17	SARAH	JOHNSON	190 E. Stacy Rd	Allen	USA	5/7/2021, 12:00:00 A	SOUTH	sarah.j@fancym
9	18	EMILY	WILSON	1168 State College B	Anaheim	USA	5/12/2021, 12:00:00	WEST	emily.w@funma

5.2-- Insert a new row into the PUR_PRODUCT table

```
INSERT INTO PUR_PRODUCT (PRODUCT_ID, PRODUCT_NAME, STANDARD_COST, LIST_PRICE)
VALUES (900, 'Chromebook', 400, 500);
-- Display the updated PUR_PRODUCT table
SELECT * FROM PUR_PRODUCT;
```

	PRODUCT_ID	PRODUCT_NAME	STANDARD_COST	LIST_PRICE
1	500	Airpods Pro	160	200
2	100	Mobile Cover	30	35
3	300	LG F100	100	15
4	400	Apple A100	110	125
5	200	Samsung F7100	80	100
6	600	MacBook	1000	1200
7	700	Sony A9	200	225
8	900	Chromebook	400	500

5.3

```
-- Modify the datatype of the TOTAL_AMOUNT column in PUR_SALES table
ALTER TABLE PUR_SALES
MODIFY TOTAL_AMOUNT NUMBER(15,2);
-- Describe the PUR_SALES table to verify the changes
DESC PUR_SALES;
```

```
Name         Null?    Type
------------ -------- ------------
SALES_ID     NOT NULL NUMBER(38)
SALES_DATE            DATE
ORDER_ID              NUMBER(38)
PRODUCT_ID            NUMBER(38)
CUSTOMER_ID           NUMBER(38)
UNIT_PRICE            NUMBER(10,2)
DISCOUNT              NUMBER(10,2)
TOTAL_AMOUNT          NUMBER(15,2)
```

6.1

SELECT PRODUCT_ID, PRODUCT_NAME, LIST_PRICE

FROM PUR_PRODUCT

WHERE LIST_PRICE >= 100;

	PRODUCT_ID	PRODUCT_NAME	LIST_PRICE
1	200	Samsung F7100	100
2	400	Apple A100	125
3	500	Airpods Pro	200
4	600	MacBook	1200
5	700	Sony A9	225
6	900	Chromebook	500

6.2

```
SELECT ROUND(AVG(UNIT_PRICE), 2) AS "Average Product Cost in South Region"
FROM PUR_SALES
JOIN PUR_CUSTOMER ON PUR_SALES.CUSTOMER_ID = PUR_CUSTOMER.CUSTOMER_ID
WHERE PUR_CUSTOMER.REGION = 'SOUTH';
```

	PRODUCT COST IN SOUTH
1	76.67

6.3

```
-- Get the customer ID, first name, last name, and total amount purchased by each customer
SELECT PUR_CUSTOMER.CUSTOMER_ID,
    PUR_CUSTOMER.FIRST_NAME,
```

```
    PUR_CUSTOMER.LAST_NAME,
    SUM(PUR_SALES.TOTAL_AMOUNT) AS TOTAL_AMOUNT
FROM PUR_CUSTOMER
JOIN PUR_SALES ON PUR_CUSTOMER.CUSTOMER_ID = PUR_SALES.CUSTOMER_ID
GROUP BY PUR_CUSTOMER.CUSTOMER_ID, PUR_CUSTOMER.FIRST_NAME, PUR_CUSTOMER.LAST_NAME
ORDER BY TOTAL_AMOUNT ASC;
```

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	TOTAL_AMOUNT
1	18	EMILY	WILSON	75
2	12	ALICA	EDWARD	100
3	14	DAVE	TAYLOR	125
4	15	DIANA	THOMAS	200
5	17	SARAH	JOHNSON	200
6	13	PETER	EDWARD	210
7	11	JANE	DOE	440

6.4

```
-- Get the salespersons with total commission above 13, sorted by total commission in descending order
SELECT
   PUR_SALESPERSON.SALESPERSON_ID,
   CONCAT(PUR_SALESPERSON.FIRST_NAME, ' ', PUR_SALESPERSON.LAST_NAME) AS "Salesperson",
   SUM(PUR_SALES_CONTACT.COMMISSION) AS "Commission earned"
FROM PUR_SALESPERSON
JOIN PUR_SALES_CONTACT ON PUR_SALESPERSON.SALESPERSON_ID =
PUR_SALES_CONTACT.SALESPERSON_ID
GROUP BY PUR_SALESPERSON.SALESPERSON_ID, PUR_SALESPERSON.FIRST_NAME,
PUR_SALESPERSON.LAST_NAME
HAVING SUM(PUR_SALES_CONTACT.COMMISSION) > 13
ORDER BY "Commission earned" DESC;
```

	SALESPERSON_ID	SALESPERSON	COMMISSION EARNED
1	11006	Olivia White	43.4
2	11005	David Brown	32.8
3	11007	Robert Taylor	16.6
4	11004	Tom Pardi	15

6.5

```
-- Find the salesperson with the highest total sales
SELECT
```

```
    CONCAT(PUR_SALESPERSON.FIRST_NAME, ' ', PUR_SALESPERSON.LAST_NAME) AS "Salesperson",
    SUM(PUR_SALES.TOTAL_AMOUNT) AS "Total Sales"
FROM PUR_SALESPERSON
JOIN PUR_SALES_CONTACT ON PUR_SALESPERSON.SALESPERSON_ID =
PUR_SALES_CONTACT.SALESPERSON_ID
JOIN PUR_SALES ON PUR_SALES_CONTACT.SALES_ID = PUR_SALES.SALES_ID
GROUP BY PUR_SALESPERSON.FIRST_NAME, PUR_SALESPERSON.LAST_NAME
ORDER BY "Total Sales" DESC
FETCH FIRST 1 ROWS ONLY;
```

	SALESPERSON	TOTAL SALES
1	Tom Pardi	520

7.1

```
SELECT CNUM AS "Customer Name", PRIORITY

FROM RES_CUSTOMERS

WHERE PRIORITY BETWEEN 15 AND 25

ORDER BY "Customer Name" ASC;
```

	CUSTOMER NAME	PRIORITY
1	C100	20
2	C400	20

7.2

```
-- Find the total number of distinct dishes prepared in each restaurant headquartered in Boston or Los
Angeles
SELECT
    RES_RESTAURANTS.RESTAURANTNAME,
    COUNT(DISTINCT RES_DISHES.DISHNAME) AS DISH_COUNT
FROM RES_RESTAURANTS
JOIN RES_ORDERS ON RES_RESTAURANTS.RNUM = RES_ORDERS.RNUM
JOIN RES_DISHES ON RES_ORDERS.DNUM = RES_DISHES.DNUM
WHERE RES_RESTAURANTS.HQLOCATION IN ('Boston', 'LosAngeles')
GROUP BY RES_RESTAURANTS.RESTAURANTNAME;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download | Execution time: 0.018 seconds

	RESTAURANTNAM	DISH_COUNT
1	SiamPalace	2
2	CheesecakeFactory	6
3	GoldenDragon	2

7.3

```
SELECT
   RES_DISHES.DISHNAME,
   CONCAT('$', ROUND(RES_DISHES.LISTPRICE, 0)) AS LIST_PRICE,
   RES_DISHES.CALORIES
FROM RES_ORDERS
JOIN RES_CUSTOMERS ON RES_ORDERS.CNUM = RES_CUSTOMERS.CNUM
JOIN RES_DISHES ON RES_ORDERS.DNUM = RES_DISHES.DNUM
WHERE RES_CUSTOMERS.CUSTNAME = 'Howard';
```

Captured with Xnip

Download | Execution time: 0.009 seconds

	DISHNAME	LIST_PRICE	CALORIES
1	BBQbaconBurger	$10	1200
2	MeeKrob	$25	600
3	MeeKrob	$25	600
4	ChickenSatay	$20	800
5	CashewChicken	$10	1500
6	BurritoGrande	$20	1000
7	BurritoGrande	$20	1000
8	BurritoGrande	$20	1000
9	SteamedDumpling	$10	400
10	SteamedDumpling	$10	400

7.4

```
SELECT
   RES_ORDERS.CNUM AS "Customer Number",
   SUM(RES_ORDERS.QUANT) AS "total cashew chicken consumed"
FROM RES_ORDERS
JOIN RES_DISHES ON RES_ORDERS.DNUM = RES_DISHES.DNUM
WHERE RES_DISHES.DISHNAME = 'CashewChicken'
GROUP BY RES_ORDERS.CNUM
ORDER BY "total cashew chicken consumed" DESC;
```

	CUSTOMER NUMBER	TOTAL CASHEW CHICKEN CONSUMED
1	C500	8
2	C300	5

7.5

```
SELECT
   RES_DISHES.DISHNAME,
   RES_RESTAURANTS.RESTAURANTNAME,
   ROUND(SUM(RES_ORDERS.PRICE), 0) AS "Total Purchase"
FROM RES_ORDERS
JOIN RES_DISHES ON RES_ORDERS.DNUM = RES_DISHES.DNUM
JOIN RES_RESTAURANTS ON RES_ORDERS.RNUM = RES_RESTAURANTS.RNUM
WHERE RES_DISHES.CITYOFORIGIN = 'Houston'
GROUP BY RES_DISHES.DISHNAME, RES_RESTAURANTS.RESTAURANTNAME
ORDER BY RES_DISHES.DISHNAME, RES_RESTAURANTS.RESTAURANTNAME;
```

	DISHNAME	RESTAURANTNAME	TOTAL PURCHASE
1	BBQbaconBurger	CheesecakeFactory	20
2	BBQbaconBurger	SiamPalace	9
3	CashewChicken	CheesecakeFactory	24
4	CashewChicken	SzechuanPalace	23
5	SteamedDumpling	CheesecakeFactory	4
6	SteamedDumpling	Giacomos	3
7	SteamedDumpling	GoldenDragon	5
8	SteamedDumpling	SzechuanPalace	2