Introduction to Data Structure (Data Management) Lecture 5

Felipe P. Vista IV

Reminder

- Everybody, make sure that your name in ZOOM is in the following format:
  - University ID Num Name (no “( )“)
  - Ex: 202054321 Juan Dela Cruz
  -
  - Not changing your name to this format
    - you might be marked Absent
    - * → absent?

Aggregations

- Aggregations (6.4.3 – 6.4.6)

- Grouping & Aggregations (6.4.3 – 6.4.6)

Grouping & Aggregation

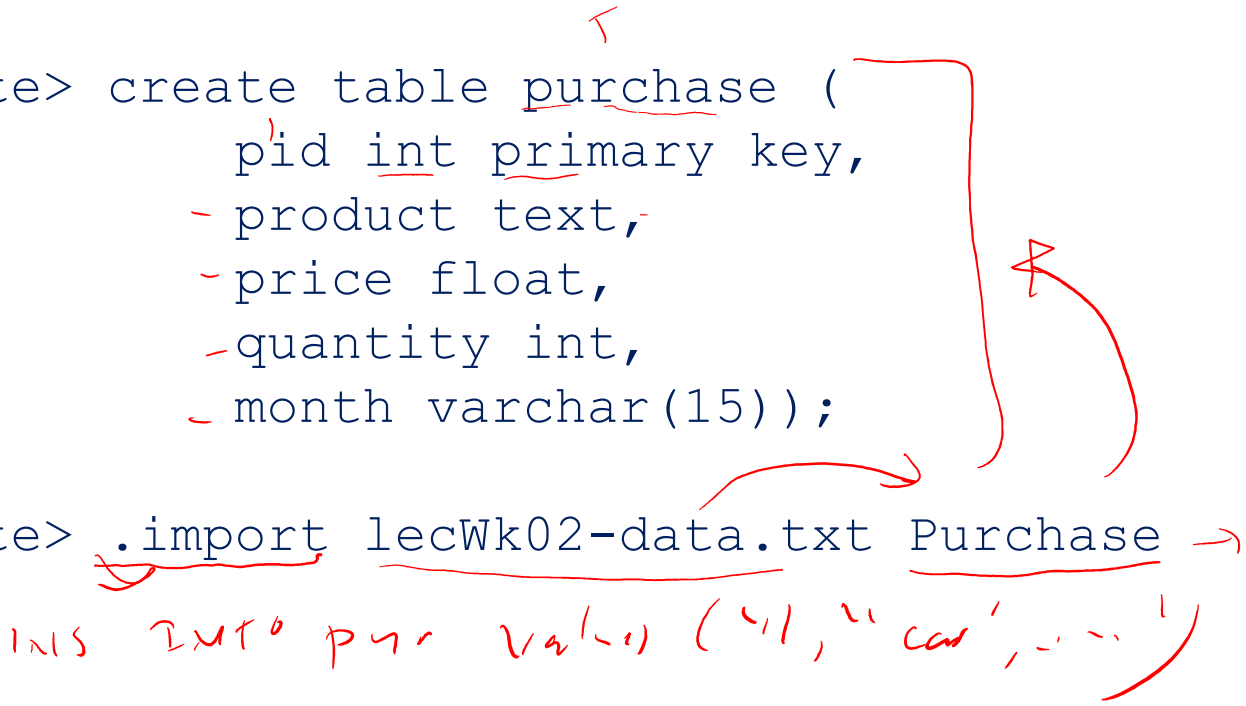
AGGREGATIONS

Aggregation in SQL

```
$>sqlite3 lecWk02

sqlite> create table purchase (
            pid int primary key,
            product text,
            price float,
            quantity int,
            month varchar(15));

sqlite> .import lecWk02-data.txt Purchase
```



lecWk02-data.txt (sample data)

PID, product, price, quantity, month

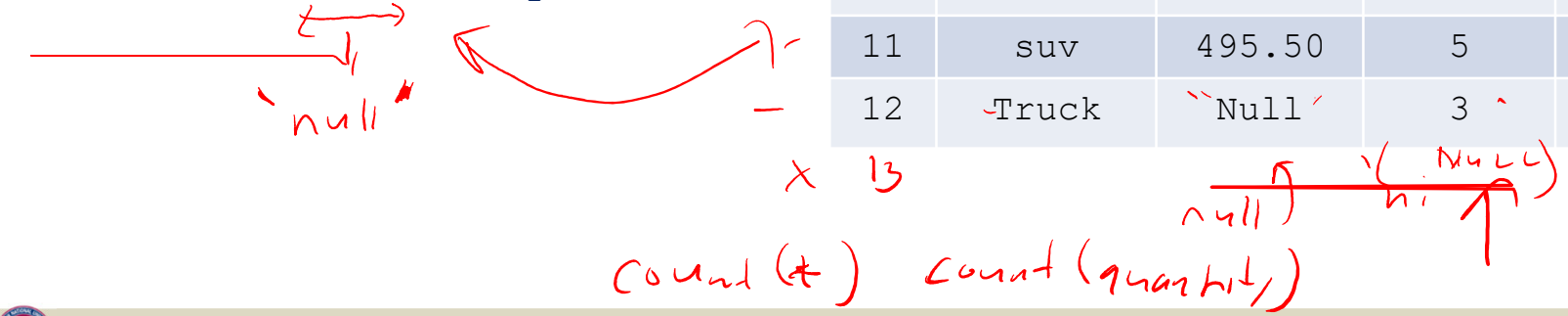
```
01|bike|119.95|20|september
02|bike|130.00|20|december
03|scooter|255.00|50|september
04|scooter|275.00|10|november
05|scooter|339.99|10|december
06|genesis|450.99|6|february
07|genesis|450.99|5|march
08|genesis|425.55|6|april
09|suv|590.99|3|january
10|suv|590.99|4|february
11|suv|495.50|5|march
12|truck|null|5|may
```

Aggregations

lecWk02-data.txt (sample data)

```
01|bike|119.95|20|september
02|bike|130.00|20|december
03|scooter|255.00|50|september
04|scooter|275.00|10|november
05|scooter|339.99|10|december
06|genesis|450.99|6|february
07|genesis|450.99|5|march
08|genesis|425.55|6|april
09|suv|590.99|3|january
10|suv|590.99|4|february
11|suv|495.50|5|march
12|truck|null|5|may
```

PID	Product	Price	Quantity	Month
01	bike	119.95	20	september
02	bike	130.00	20	december
03	scooter	255.00	50	september
04	scooter	275.00	10	november
05	scooter	339.99	10	december
06	genesis	450.99	6	february
07	genesis	450.99	5	march
08	genesis	425.55	6	april
09	suv	590.99	3	january
10	suv	590.99	4	february
11	suv	495.50	5	march
12	Truck	Null	3	may



Notes on SQLite

- Cannot directly load NULL values as NULL in the DB

Notes on SQLite

- Cannot directly load NULL values as NULL in the DB

- Two steps are needed:
  - load NULL values using some type of special value
  - upload the special values to actual NULL values

```
UPDATE purchase
   SET price = null
WHERE price = 'null'
```

Simple Aggregations

- Five basic aggregate operations in SQL

```
SELECT count(*) FROM purchase
SELECT sum(quantity) FROM purchase
SELECT avg(price) FROM purchase
SELECT max(quantity) FROM purchase
SELECT min(quantity) FROM purchase
```

Aggregates and NULL Values

- NULL values are not used in aggregates

```
INSERT INTO purchase
   VALUES(13, 'truck', NULL, NULL, 'april')
```

Aggregates and NULL Values

- NULL values are not used in aggregates

```
INSERT INTO purchase
   VALUES(13, 'truck', NULL, NULL, 'april')
```

- Trying the following

```
SELECT count(*) FROM purchase

SELECT count(quantity) FROM purchase

SELECT sum(quantity) FROM purchase

SELECT sum(quantity)
   FROM purchase
WHERE quantity IS NOT NULL
```

Aggregates and NULL Values

- NULL values are not used in aggregates

```
INSERT INTO purchase
    VALUES(13, 'truck', NULL, NULL, 'april')
```

- Trying the following

```
SELECT count(*) FROM purchase
```

→ NULL is counted in count(*)

```
SELECT count(quantity) FROM purchase
```

→ NULL is ignored in count(quantity)

```
SELECT sum(quantity) FROM purchase
```

```
SELECT sum(quantity)
    FROM purchase
WHERE quantity IS NOT NULL
```

→ "IS NOT NULL" is redundant

To 48

Counting Duplicates

- COUNT applies to duplicates, unless otherwise stated:

```
SELECT count(product)
    FROM purchase
WHERE price>300.00
```

→ Same as count(*) if there are no NULLS

bike
bike
scooter
sky
sky

Counting Duplicates

- COUNT applies to duplicates, unless otherwise stated:

```
SELECT count(product)
    FROM purchase
WHERE price> 300.00
```

→ Same as count(*) if there are no NULLS

bike - 2
scooter - 3
genesis - 3
suv - 3
truck - 1

- Or maybe want to:

```
SELECT count(DISTINCT product)
    FROM purchase
WHERE price> 300.00
```

bike
scooter
suv
genesis
truck

Counting Duplicates (More Examples)

```
SELECT sum(price * quantity)
    FROM purchase
```

```
SELECT sum(price * quantity)
    FROM purchase
WHERE product='bike'
```

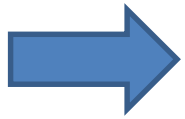
What does this mean?

Simple Aggregations

Purchase

product	price	quantity
bike	119.95	20
bike	130.00	20
scooter	255.00	50
scooter	275.50	10
scooter	339.99	10

```
SELECT sum(price * quantity)
    FROM purchase
WHERE product='bike'
```



```
(119.95x20)+
(130.00x20) =
 4999.00
```

More Examples

- How to get the average revenue per sale?

```
SELECT sum(price*quantity)/count(*)
    FROM purchase
WHERE product='bike'
```

More Examples

- How to get the average revenue per sale?

```
SELECT sum(price*quantity)/count(*)
    FROM purchase
WHERE product='bike'
```

- How to get the average price of a bike sold?

```
SELECT sum(price*quantity)/sum(quantity)
    FROM purchase
WHERE product='bike'
```

More Examples

- How to get the average revenue per sale?

```
SELECT sum(price*quantity)/count(*)
    FROM purchase
WHERE product='bike'
```

- How to get the average price of a bike sold?

```
SELECT sum(price*quantity)/sum(quantity)
    FROM purchase
WHERE product='bike'
```

- What happens if there are NULLs in price or quantity?
  - It might affect the result
  - Lesson? Do not allow NULL unless really needed

More Examples

- How to get the average revenue per sale?

```
SELECT sum(price*quantity)/count(*)
    FROM purchase
WHERE product='bike'
```

More Examples

- How to get the average revenue per sale?

```
SELECT sum(price*quantity)/count(*)
    FROM purchase
WHERE product='bike'
```

- How to get the average price of a bike sold?

```
SELECT sum(price*quantity)/sum(quantity)
    FROM purchase
WHERE product='bike'
```

Grouping & Aggregation

GROUPING & AGGREGATIONS

Aggregation

- Purchase(product, price, quantity)
- How many bikes sold?

product	price	quantity
bike	119.95	20
bike	130.00	20
scooter	255.00	50
scooter	275.50	10
scooter	339.99	10

```
SELECT sum(quantity) AS TotalSold
   FROM purchase
WHERE price ≥ 250 and product = 'bike'
```

Grouping & Aggregation

- Purchase(product, price quantity)
- How many bikes sold?

product	price	quantity
bike	119.95	20
bike	130.00	20
scooter	255.00	50
scooter	275.50	10
scooter	339.99	10

```
SELECT sum(quantity) AS TotalSold
    FROM purchase
WHERE price > 125.00
GROUP BY product
```

What does this mean?

Grouping & Aggregation

1. Compute FROM and WHERE clauses

2. Group by the attributes through GROUP BY

3. Compute the SELECT clause:
   - Grouped attributes and aggregates

Grouping & Aggregation

1. Compute FROM and WHERE clauses

2. Group by the attributes through GROUP BY

3. Compute the SELECT clause:
   - Grouped attributes and aggregates

FWGS

Grouping & Aggregation

- Steps 1 & 2 FROM-WHERE-GROUP BY

FWGS

Product	Price	Quantity
bike	119.95	20
bike	130.00	20
scooter	255.00	50
scooter	275.50	10
scooter	339.99	10

```
WHERE
price > 125.00
```

Grouping & Aggregation

- Step 3 SELECT

FWGS

Product	Price	Quantity
bike	119.95	20
bike	130.00	20
scooter	255.00	50
scooter	275.50	10
scooter	339.99	10

Product	Sum(Quantity)
bike	20
scooter	70

```
SELECT sum(quantity) AS TotalSold
    FROM purchase
WHERE price > 125.00
GROUP BY product
```

Other Examples

```
Purchase(pid,product,price,quantity,month)
```

Compare these two queries:

```
SELECT   product, count(*)
  FROM   purchase
GROUP BY product
```

```
SELECT   month, count(*)
  FROM   purchase
GROUP BY month
```

Other Examples

Purchase(pid,product,price,quantity,month)

Compare these two queries:

```
SELECT   product, count(*)
   FROM  purchase
GROUP BY product
```

```
SELECT   month, count(*)
   FROM  purchase
GROUP BY month
```

How about this:

```
SELECT   product,
         sum(quantity) AS sumQuantity,
         max(price) AS maxPrice,
   FROM  purchase
GROUP BY product
```

Other Examples

PID	Product	Price	Quantity	Month
01	bike	119.95	20	september
02	bike	130.00	20	december
03	scooter	255.00	50	september
04	scooter	275.00	10	november
05	scooter	339.99	10	december
06	genesis	450.99	6	february
07	genesis	450.99	5	march
08	genesis	425.55	6	april
09	suv	590.99	3	january
10	suv	590.99	4	february
11	suv	495.50	5	march
12	Truck	Null	3	may
13	Truck	Null	Null	April

Need to be Careful

```
Purchase(pid,product,price,quantity,month)
```

```
SELECT product, max(quantity)
       FROM purchase
GROUP BY product
```

```
SELECT product, quantity
   FROM purchase
GROUP BY month
```

Product	Price	Quantity
bike	119.95	20
bike	130.00	20
scooter	255.00	50
scooter	275.50	10
scooter	339.99	10

sqlite allows this query but with strange results

Better DBMS has error trap (ex: SQL Server)

ASSIGNMENT #02

- Download Assign02.sql file
- Rename Assign02.sql to Assign02-UniIDNum.sql
  - Ex: Assign02.sql → Assign02-202012345.sql
- Follow the instructions & answer questions inside Assign02.sql file by providing correct SQL Statements
- Check your answer by:
  - Run SQLite on your own database, i.e. 202012345
    - `C:\sqlite3 202012345`
  - Run your answers in sql file
    - `sqlite3> .read Assign02-202012345.sql`
  - Results must be similar to **Assign02-results.jpg**

Thank you.