Here's a comprehensive outline and HiveQL code snippets to help you build an application that uses **HBase** and **Hive** for querying the **Online Retail Dataset**. The dataset typically includes fields like `InvoiceNo`, `StockCode`, `Description`, `Quantity`, `InvoiceDate`, `UnitPrice`, `CustomerID`, and `Country`.

Step i: Create and Load Table with Online Retail Data in Hive

Assuming your data is in a CSV format and has been uploaded to HDFS:

sql
CopyEdit

```
CREATE TABLE online_retail_raw (
    InvoiceNo STRING,
    StockCode STRING,
    Description STRING,
    Quantity INT,
    InvoiceDate STRING,
    UnitPrice DOUBLE,
    CustomerID STRING,
    Country STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- Load data into Hive table
LOAD DATA INPATH '/user/hive/warehouse/online_retail.csv' INTO
TABLE online_retail_raw;
```

Step j: Create Index on Online Retail Table in Hive

Hive supports indexing, though it is less commonly used due to performance overhead in newer versions. Still, for academic/demo purposes:

sql
CopyEdit

```
CREATE INDEX idx_customerid
ON TABLE online_retail_raw (CustomerID)
AS 'COMPACT'
WITH DEFERRED REBUILD;

ALTER INDEX idx_customerid ON online_retail_raw REBUILD;
```

> 📝 Note: Indexes in Hive can help in certain query optimizations but are not widely used in modern big data processing where partitioning and bucketing are more efficient.

Step k: Find the Total, Average Sales in Hive

To compute sales, we'll use the formula:

```
Sales = Quantity * UnitPrice
```

sql
CopyEdit

```
SELECT
    SUM(Quantity * UnitPrice) AS total_sales,
    AVG(Quantity * UnitPrice) AS average_sales
FROM online_retail_raw;
```

Step l: Find Order Details with Maximum Cost

sql
CopyEdit

```
SELECT InvoiceNo, SUM(Quantity * UnitPrice) AS order_total
FROM online_retail_raw
GROUP BY InvoiceNo
ORDER BY order_total DESC
LIMIT 1;
```

Step m: Find Customer Details with Maximum Order Total

sql
CopyEdit

```
SELECT CustomerID, SUM(Quantity * UnitPrice) AS customer_total
FROM online_retail_raw
WHERE CustomerID IS NOT NULL
GROUP BY CustomerID
ORDER BY customer_total DESC
LIMIT 1;
```

Optional: Integrate with HBase (for advanced use)

If you want to store Hive table data in HBase for fast key-based access, you can create an external Hive table backed by HBase:

sql
CopyEdit

```
CREATE TABLE hbase_online_retail(
    rowkey STRING,
    InvoiceNo STRING,
    StockCode STRING,
    Description STRING,
    Quantity INT,
    InvoiceDate STRING,
    UnitPrice DOUBLE,
    CustomerID STRING,
    Country STRING
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =

":key,cf1:InvoiceNo,cf1:StockCode,cf1:Description,cf1:Quantity,c
f1:InvoiceDate,cf1:UnitPrice,cf1:CustomerID,cf1:Country")
```

```
TBLPROPERTIES ("hbase.table.name" = "hbase_online_retail");
```

Then use Hive `INSERT INTO hbase_online_retail SELECT ...` to populate the HBase-backed table.